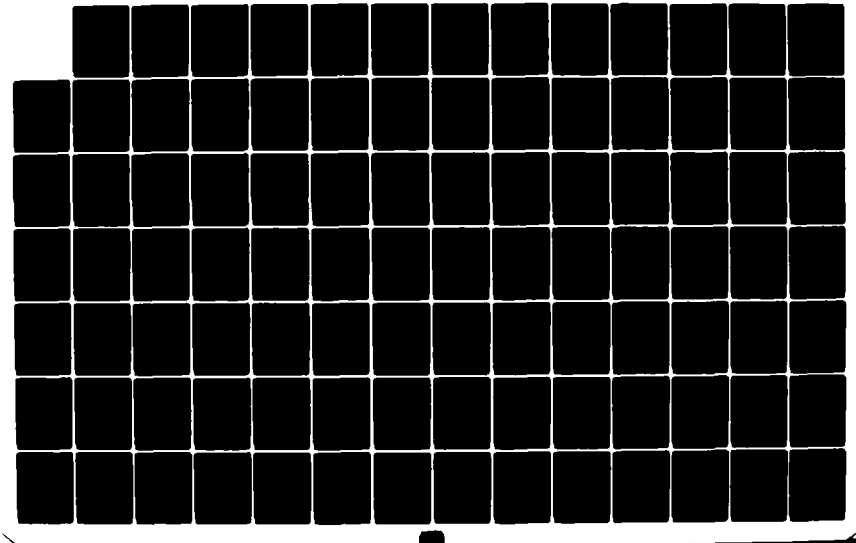
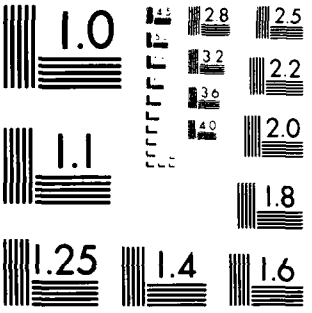
AD-A127 689 A BIBLIOGRAPHY OF SOFTWARE ENGINEERING TERMS(U) DATA 1/2
AND ANALYSIS CENTER FOR SOFTWARE GRIFFISS AFB NY
S A GLOSS-SOLER OCT 79 DACS-GLOS-1 F30602-78-C-0255
UNCLASSIFIED F/G 5/2 NL



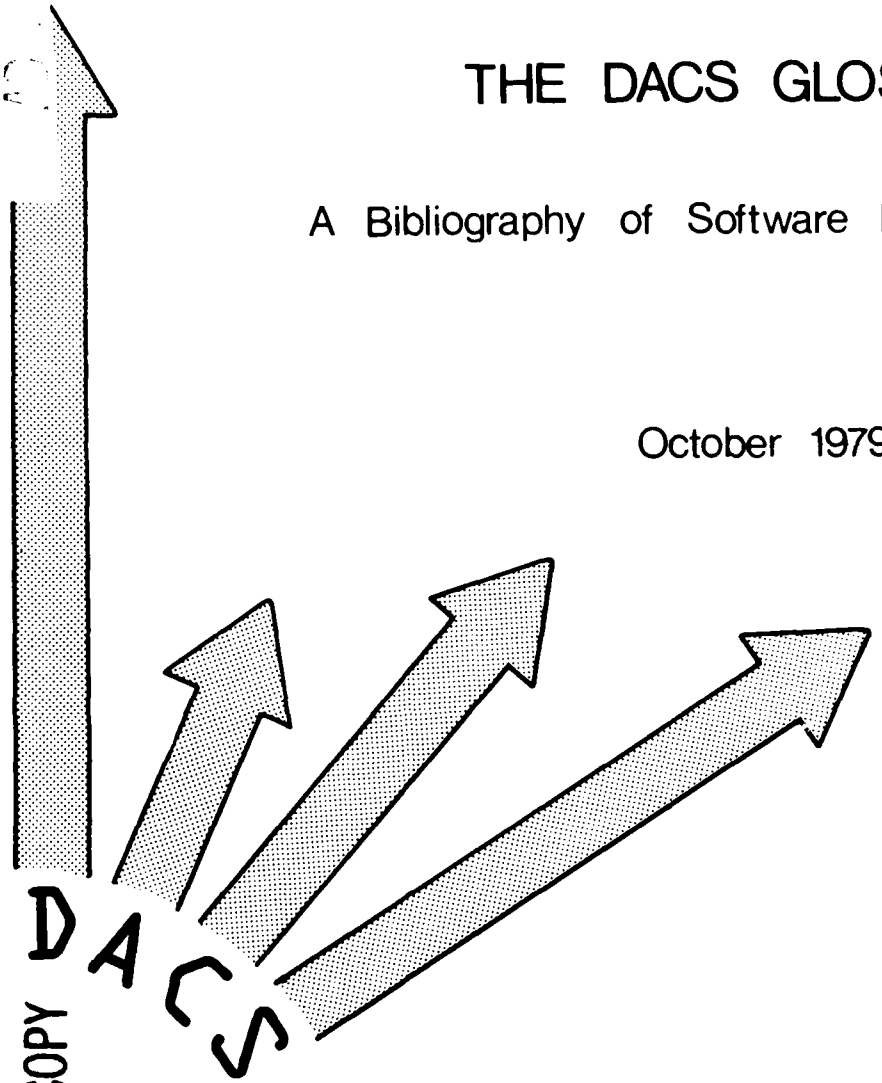


GLOS-1

THE DACS GLOSSARY

A Bibliography of Software Engineering Terms

October 1979



Data & Analysis Center for Software

DACS **Data & Analysis Center for Software**

AN INFORMATION ANALYSIS CENTER

THE DACS GLOSSARY

A Bibliography of Software Engineering Terms

COMPILED FROM THE LITERATURE

BY:

SHIRLEY A. GLOSS-SOLER
IIT RESEARCH INSTITUTE

UNDER CONTRACT TO:

ROME AIR DEVELOPMENT CENTER
GRIFFISS AIR FORCE BASE

OCTOBER 1979

ORDERING NUMBER GLOS-1

DACS

The **Data & Analysis Center for Software** is an Information Analysis Center, operated by IIT Research Institute under contract to the Rome Air Development Center, AFSC.

The Data and Analysis Center for Software (DACS) is an information analysis center sponsored by the Air Force Systems Command, Rome Air Development Center (RADC), and operated by IIT Research Institute (IITRI). DACS serves as a central source for current, readily usable data and information concerning software technology.

The major functions of the DACS are to develop and maintain a computer database of empirical data collected on the development and maintenance of computer software; produce and distribute subsets of the database; maintain a software technology information base of technical documents, project status information, and evaluation data; analyze the data and information and produce technical reports; maintain a current awareness program which includes dissemination of technical information (analysis reports, technical monographs, etc.), assessments of technological developments, and publication of a quarterly newsletter; develop and maintain a glossary of software engineering terms; and provide rapid response to inquiries for technical information and assistance.

To obtain more information on the products and services of the DACS, contact:

Lorraine Duvall
Data and Analysis Center for Software
RADC/ISISI
Griffiss Air Force Base, NY 13441

Telephone: 315/336-0937
Autovon: 587-3395

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER GLOS-1	2. GOVT ACCESSION NO. AD-A127689	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* THE DACS GLOSSARY A Bibliography of Software Engineering Terms		5. TYPE OF REPORT & PERIOD COVERED Interim Report Sept.-78 - Aug. 79 6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Shirley A. Gloss-Soler		8. CONTRACT OR GRANT NUMBER(s) F30602-78-C-0255
9. PERFORMING ORGANIZATION NAME AND ADDRESS Data & Analysis Center for Software RADC/ISISI Griffiss AFB, NY 13441		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEE) Griffiss AFB, NY 13441		12. REPORT DATE October 1979 13. NUMBER OF PAGES 147
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* Same		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John Palaimo (COEE)

~~Available from: Defense Technical Information Center~~

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Software Engineering Terminology
Software Technology
Computer Software
Software Engineering

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The DACS Glossary contains over 1100 terms and their definitions compiled from the software engineering literature. Sources for definitions are cited and ordering information for source documents is supplied.

DD FORM 1 JAN 73 1473

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

PREFACE

The purpose of this document is to record, as accurately as is possible in a still-evolving discipline, the terminology currently being used in the field of software engineering. We hope that the DACS GLOSSARY will help to improve communication within the software engineering community and will also provide an impetus toward the sorely needed standardization of terminology.

This software engineering glossary is one of the products of the Data and Analysis Center for Software (DACS). The DACS will continue to update this glossary to reflect current term usage. Suggestions, comments, and critiques are welcome.

A

IV

THE DACS GLOSSARY

A BIBLIOGRAPHY OF SOFTWARE ENGINEERING TERMS

Table of Contents

SECTION I

INTRODUCTION

1.1 Contents

This document contains definitions of terms from the software engineering literature and consists of three sections. This first section provides introductory information for the users of the glossary. The second section contains the terms, their definitions, and a reference to the source which supplied the definition. Of the 1123 terms included, 1100 have one or more definitions and 23 are cross references to other terms. Much of the terminology currently in use is not used consistently and, often, no generally perferred definition has yet emerged; for these terms we have included alternate definitions. In these cases, the first (not necessarily authoritative) definition is followed by its reference in parentheses, and the reference is followed by a numeral in parentheses, then by the alternate definition itself. The same notational method is used for successive alternate definitions. The third section lists the 171 sources referenced. The third section also provides ordering information for the source documents.

Terms and definitions have been obtained from many sources; from software engineering literature, from lists contributed by various individuals, and from published dictionaries of data processing terminology. These sources have been credited by several codes which are identified on the pages immediately following the listing of terms. Several definitions are a synthesis of more than one definition and cannot be credited to a single source.

Two sources require special mention. Those terms credited to ANSI-X3 are taken from the American National Dictionary for Information Processing. Complete copyright and purchase information for the dictionary is supplied in the list of sources. Those terms credited to ANSI-X3H1 were compiled by the Standing Committee on Operating System Command Languages of the American National Standards Institute.

1.2 Objectives

There were two objectives in compiling this software engineering glossary. The first, as stated in the preface, is to record the terminology currently in use. The second objective is to provide users of the DACS software engineering bibliographic services [1] a means of ensuring that they are using terms from the DACS THESAURUS (a specialized thesaurus for indexing and retrieving software engineering literature) for their retrievals in the same way as the terms were used for indexing. A closely related objective is to promote consistency in indexing new documents for the bibliographic collection.

Certain terms are contained in this glossary which are not specific to computer software--or even to hardware--but which describe concepts of interest to users of the bibliographic information database (e.g., Submarine Applications). These terms are identified as "Indexing Terms" and are not defined; instead a description is given of the type of information a document retrieved using that term would contain.

A draft version of this glossary has already been used by the Software Engineering Terminology Task Group [2] as a base from which candidate terms were selected for a working draft of the second version of their Software Engineering Terminology. The results of their efforts will help us to make an improved version of the DACS Glossary for later publication. We welcome the opportunity to participate in such reciprocal efforts.

[1] More information is contained in the DACS publication "Bibliographic Services - Custom Searches", order no. BIB-1. BIB-1 also contains the DACS Thesaurus.
[2] IEEE Computer Society, Technical Committee on Software Engineering, Subcommittee on Software Engineering Standards, Terminology Task Group.

SECTION II

TERMS AND DEFINITIONS

ABSOLUTE MACHINE CODE
MACHINE LANGUAGE CODE IN WHICH ADDRESSES ARE SPECIFIED IN TERMS OF ACTUAL MACHINE LOCATIONS. (NASA)

ABSTRACT MACHINES
FICTITIOUS COMPUTERS USED TO IMPLEMENT PORTABLE OPERATING SYSTEMS. (DAN 284) (2) A LOGICAL ENTITY COMPOSED OF: 1) SPECIFIC, FIXED DATA OBJECTS; 2) A FIXED COLLECTION OF DATA OBJECT CLASSES CALLED DATA TYPES; 3) A FIXED COLLECTION OF OPERATIONS; AND, OPTIONALLY, 4) AN ONGOING MACHINE CYCLE. THE CAPABILITIES AND CHARACTERISTICS OF AN ABSTRACT MACHINE MAY BE FULLY DESCRIBED IN TERMS OF THE DATA OBJECTS, DATA TYPES, AND OPERATIONS AND MACHINE CYCLE WHICH MAKE UP THE MACHINE. (ABBOTT) (3) A REPRESENTATION OF THE CHARACTERISTICS OF A MACHINE AS SEEN BY A USER OR PROGRAM. (ANSI-X3H1)

ABSTRACT RESOURCE
ANY COMMODITY OR AVAILABLE MEANS THAT MAY BE ALLOCATED TOWARD THE ACCOMPLISHMENT OF A TASK, CHARACTERIZABLE BY ABSTRACTIONS IN REPRESENTATION, MANIPULATIONS, AND AXIOMATIZATION. (DAN 1153)

ABSTRACTION
A MECHANISM FOR HIERARCHIC, STEPWISE REFINEMENT OF DETAIL BY WHICH IT IS POSSIBLE AT EACH STAGE OF DEVELOPMENT TO EXPRESS ONLY RELEVANT DETAILS AND TO DEFER (AND, INDEED, HIDE) NON-RELEVANT DETAILS FOR LATER REFINEMENT. (DAN 1153)

ACCEPTANCE CRITERIA
CRITERIA THAT A SET OF SOFTWARE MUST SATISFY IN CONFORMANCE WITH DELIVERY REQUIREMENTS. SOFTWARE DELIVERED FOR INTERIM OPERATIONS WITH DISCREPANT ITEMS IS SAID TO BE ACCEPTED WITH "LIENS". (DAN 1153)

ACCEPTANCE TESTING
TESTING TO VERIFY ACCEPTANCE CRITERIA FOR PROGRAM CERTIFICATION. (DAN 1153) (2) THIS IS A SELF-DEFINING TERM UTILIZED IN SOFTWARE AND/OR HARDWARE PRODUCING CONTRACTS. THE PRODUCT'S PASS/FAIL CRITERIA ARE PREDETERMINED. FAILURE TO MEET THE STANDARD OF THE CRITERIA CAUSES REJECTION OF THE PRODUCT.

ACCESSIBILITY
CODE POSSESSES THE CHARACTERISTIC ACCESSIBILITY TO THE EXTENT THAT IT FACILITATES SELECTIVE USE OF ITS PARTS. (ACCESSIBILITY IS NECESSARY FOR EFFICIENCY, TESTABILITY, AND HUMAN ENGINEERING) (DAN 239) EASE OF ACCESS TO A SYSTEM. ACCESSIBILITY IS A REFLECTION OF THE PROBABILITY OF INTENTIONAL AND ACCIDENTAL BREAKING INTO A SYSTEM. ACCESSIBILITY IS NEARLY SYNONOMOUS WITH SECURITY. (DAN 781)

ACCESS-CONTROL MECHANISMS
ACCESS CONTROL MECHANISMS ARE MECHANISMS CAPABLE OF ENFORCING RULES ABOUT WHO CAN PERFORM WHAT OPERATIONS OR WHO CAN ACCESS AN OBJECT CONTAINING CERTAIN INFORMATION. (DAN 616)

ACCREDITATION
ALL ACTIVITIES THAT, TAKEN TOGETHER, ESTABLISH A SUFFICIENT LEVEL OF CONFIDENCE IN THE FINAL PRODUCT THAT THE DEVELOPER IS ABLE TO GUARANTEE ITS FUNCTIONAL PERFORMANCE TO SPECIFICATIONS AND TO PROVIDE A WARRANTY TO THE

CUSTOMER WITH MINIMUM RISK OF ADDITIONAL SUPPORT AT THE DEVELOPER'S EXPENSE. (DAN-LD7) (2) ACCREDITATION IS THE PROCESS WHEREBY ACCURACY TO A PREDEFINED STANDARD IS ASCERTAINED AND DEMONSTRATED FOR A SOFTWARE PRODUCT... ACCREDITATION AS A TERM IN SOFTWARE ENGINEERING HAS COME TO BE USED ONLY RECENTLY, PRIMARILY IN THE DOD COMMUNITY TO DESCRIBE AN AUTHORITATIVE ENDORSEMENT OF A SOFTWARE PRODUCT. IT HAS BEEN USED SYNONYMOUSLY WITH THE TERM CERTIFICATION IN THE DOD COMMUNITY. ACCREDITATION REQUIRES USER EXPERIENCE TO EVALUATE THE RELIABILITY OF THE PRODUCT. PROCEDURES FOR DIRECTLY EXAMINING THE SOFTWARE ARE ALSO REQUIRED BEFORE THE ACCREDITATION CAN BE MADE FOR THE PRODUCT IN QUESTION. (SET)

ACCURACY
A MEASURE OF THE DEGREE OF FREEDOM FROM ERROR; THE DEGREE OF EXACTNESS POSSESSED BY AN APPROXIMATION OR MEASUREMENT. IN THIS CONTEXT ACCURACY IS A MEASURE OF "DESIGN ADEQUACY" RATHER THAN "SYSTEM RELIABILITY". ERRORS WHICH INFLUENCE THIS MEASURE ARE DUE TO THE DATA AND THE LOGIC FOR PROCESSING THAT DATA. ADDITIONAL ERROR DUE TO HARDWARE FAILURE OR LOGIC "BUGS" IS HANDLED AND MEASURED SEPARATELY UNDER RELIABILITY CONCEPTS. (DAN 781)

ACCURACY STUDY PROCESSOR
A COMPUTER PROGRAM USED TO PERFORM CALCULATIONS TO ASSIST IN DETERMINING IF PROGRAM VARIABLES ARE COMPUTED WITH REQUIRED ACCURACY. (DAN 134)

ACQUISITION
THE PROCESS OF ACQUIRING SOFTWARE SYSTEMS AND/OR COMPONENTS. ACTIVITIES MAY INCLUDE DEFINING THE NEED, RESEARCHING AVAILABLE ALTERNATIVES, EVALUATING COMPETING PROPOSALS, SELECTING OR CONTRACTING FOR THE SYSTEM/COMPONENT TO BE ACQUIRED, ETC.

ACQUISITION MANAGEMENT
ALL ACTIVITIES CONDUCTED BY THE ACQUIRING ORGANIZATION TO INSURE THAT THE SOFTWARE SYSTEM OR COMPONENT BEING ACQUIRED IS DEVELOPED IN ACCORDANCE WITH ITS REQUIREMENTS.

ACTIVATE
SYNONYM FOR INVOKE (ANSI-X3H1)

ACTUAL DATA
DATA DESCRIBING THE RESULTS OF PROGRAMMING ACTIONS FOR A PROJECT THAT WILL BE THE PRIMARY DATA INCLUDED IN THE MANAGEMENT REPORTS. (DAN 137)

ADA
HIGHER ORDER PROGRAMMING LANGUAGE ORIENTED TO COMMAND AND CONTROL USE. SEE ALSO: DOD COMMON HIGH ORDER LANGUAGE

ADAPTABILITY
ADAPTABILITY IS A MEASURE OF THE EASE WITH WHICH A PROGRAM CAN BE ALTERED TO FIT DIFFERING USER IMAGES AND SYSTEM CONSTRAINTS. (DAN 758) (2) CODE POSSESSES THE CHARACTERISTIC ADAPTABILITY TO THE EXTENT THAT IT CAN BE EASILY ALTERED TO FIT DIFFERING USER IMAGES AND SYSTEM CONSTRAINTS. (DAN 239)

ADAPTION
MODIFICATION OF EXISTING SOFTWARE IN ORDER THAT IT MAY BE USED AS A MODULE

IN A PROGRAM DEVELOPMENT, AS OPPOSED TO DEVELOPING ANOTHER MODULE FOR THAT SAME PURPOSE. (DAN 1153)

ADAPTIVE TESTING
THE GOAL OF ADAPTIVE TESTING IS TO PROVIDE AN EFFECTIVE MEANS TO IDENTIFY THE BOUNDARY OF A BALLISTIC MISSILE DEFENSE SOFTWARE IMPLEMENTATION. THE PERFORMANCE BOUNDARY WILL BE REACHED BY SYSTEMATICALLY PERTURBING THE THREAT SCENARIO IN SUCH A WAY AS TO DEGRADE THE SYSTEM PERFORMANCE. (DAN 428)

AED PROGRAMMING LANGUAGE
(AUTOMATED ENGINEERING DESIGN) A HIGHER ORDER PROGRAMMING LANGUAGE BASED ON ALGOL. (NASA)

AIRBORNE WARNING AND CONTROL SYSTEM (AWACS)
A BOEING 707-320 CONVERTED TO AN AIR FORCE E-3A, USES AN AIRBORNE RADAR PLATFORM AND IS LOADED WITH THE LATEST COMMUNICATIONS, RADAR, AND COMPUTER EQUIPMENT, ENABLING THE E-3A CREW TO PERFORM COMMAND AND CONTROL SUPPORT FOR A WIDE RANGE OF MISSIONS. (DAN 385)

ALGOL
(ALGORITHMIC LANGUAGE) A BLOCK-STRUCTURED HIGH LEVEL PROGRAMMING LANGUAGE USED TO EXPRESS COMPUTER PROGRAMS BY ALGORITHMS.

ALGORITHM
A COLLECTION OF OPERATIONS ORGANIZED TO BE PERFORMED IN A CERTAIN ORDER WHEN APPLIED TO DATA OBJECTS. THE ARRANGEMENT OF THE OPERATIONS MAY LEAD TO SOME OF THE OPERATIONS BEING PERFORMED MULTIPLE TIMES AND OTHERS NOT BEING PERFORMED AT ALL. THE SELECTION AND ORDERING OF THE PERFORMANCE OF THE OPERATIONS MAY DEPEND IN PART ON THE DATA OBJECTS TO WHICH THE ALGORITHM IS APPLIED. IF AN ALGORITHM IS APPLIED TWICE TO THE SAME DATA OBJECT, THE OPERATIONS WILL BE PERFORMED IN THE SAME ORDER (YIELDING THE SAME RESULTS). THE ARRANGEMENT OF THE OPERATIONS OF AN ALGORITHM WHICH DETERMINES THEIR SELECTION AND ORDER OF PERFORMANCE IS INDICATED BY THE CONTROL STRUCTURES (AND CONTROL STATEMENTS) USED TO DEFINE THE ALGORITHM. AN ALGORITHM MAY BE USED TO DEFINE AN OPERATION (ON ONE LEVEL OF ABSTRACTION) IN TERMS OF OTHER OPERATIONS (ON A LOWER LEVEL OF ABSTRACTION). (ABBOTT) (2) A PRESCRIBED SET OF WELL-DEFINED RULES OR PROCESSES FOR THE SOLUTION OF A PROBLEM IN A FINITE NUMBER OF STEPS. IN PRINCIPLE, THE STEPS ARE SUFFICIENTLY BASIC AND DEFINITE THAT A HUMAN CAN COMPUTE ACCORDING TO THE PRESCRIBED STEPS EXACTLY AND IN A FINITE LENGTH OF TIME, USING PENCIL AND PAPER. (DAN 1153) CONTRAST WITH HEURISTIC.

ALGORITHM ANALYSIS
THE COMPARISON OF DIFFERENT ALGORITHMS AVAILABLE FOR THE ACCOMPLISHMENT OF A GIVEN TASK WITH THE PURPOSE OF SELECTING THE ONE ALGORITHM WHICH IS OPTIMAL WITH RESPECT TO TIME, SPACE, AND PERFORMANCE REQUIREMENTS. THE INPUTS TO THE ANALYSIS PROCESS ARE THE SET OF ALL DATA VALUES IN THE DOMAIN OF THE ALGORITHM, THE NUMBER OF TIMES OPERATIONS CRITICAL TO THE ALGORITHM MUST BE PERFORMED, AND THE EXPECTED VS. WORST CASE TO BE ENCOUNTERED. THE OUTPUTS ARE SPACE (MEMORY) UTILIZATION, RUNNING TIMES, AND AVERAGE VS. WORST CASE FIGURES FOR BOTH SPACE AND TIME FOR EACH ALGORITHM UNDER ANALYSIS.

ALGORITHM DESIGN
THE PROCESS OF SELECTING A "BEST KNOWN" ALGORITHM ACCORDING TO A BALANCED

SET OF ALGORITHM CHARACTERISTICS WHICH BEST SATISFY THE REQUIREMENTS OF THE SITUATION IN WHICH THE ALGORITHM IS TO PERFORM.

ALIAS
AN ADDITIONAL NAME BY WHICH AN ITEM IS KNOWN. SEE ALSO SYNONYM (ANSI-X3H1) (2) AN ADDITIONAL NAME BY WHICH A MEMORY LOCATION CAN BE REFERENCED.

ALLOCATE
TO APPORTION TO PARTICULAR PERSONS OR THINGS. TO SET APART OR EARMARK. (ANSI-X3H1)

AMBIGUOUS
CAPABLE OF BEING UNDERSTOOD IN 2 OR MORE SENSES. (ANSI-X3H1)

ANALYTICAL MODELING
THE TECHNIQUE USED TO EXPRESS MATHEMATICALLY (USUALLY BY A SET OF EQUATIONS) A REPRESENTATION OF SOME REAL PROBLEM. SUCH MODELS ARE VALUABLE FOR ABSTRACTING THE ESSENCE OF THE SUBJECT OF INQUIRY. BECAUSE EQUATIONS DESCRIBING COMPLEX SYSTEMS TEND TO BECOME COMPLICATED AND OFTEN IMPOSSIBLE TO FORMULATE, IT IS USUALLY NECESSARY TO MAKE SIMPLIFYING ASSUMPTIONS WHICH MAY DISTORT ACCURACY. SPECIFIC LANGUAGE AND SIMULATION SYSTEMS MAY SERVE AS AIDS TO IMPLEMENTATION. (DAN 134)

ANALYZER
AN ANALYZER IS A COMPUTER PROGRAM WHICH IS APPLIED TO ANOTHER PROGRAM TO PROVIDE ANALYTICAL INFORMATION. AN ANALYZER BREAKS THE PROGRAM INTO IDENTIFIABLE SMALL PARTS CALLED SEGMENTS, AND USES THE RESULTING SEGMENTS TO PRODUCE STATISTICAL INFORMATION. THIS INFORMATION CAN INCLUDE EXECUTION FREQUENCY STATISTICS, PROGRAM PATH ANALYSIS, AND/OR SOURCE CODE SYNTAX ANALYSIS. AN ANALYZER MAY BE USED TO DETERMINE (1) THE DEGREE TO WHICH TEST CASES EXERCISE THE STRUCTURE OF THE PROGRAM; (2) WHICH PROGRAM SEGMENTS ARE NOT EXECUTED; (3) WHICH SEGMENTS ARE HEAVILY EXECUTED (AND THUS ARE CANDIDATES FOR OPTIMIZATION); (4) WHICH TEST CASES NEED TO BE RERUN IF A PROGRAM SEGMENT IS CHANGED. (SET) (2) A COMPUTER PROGRAM USED TO PROVIDE SOURCE LANGUAGE OR EXECUTION FREQUENCY STATISTICS AT THE PROGRAM OR SOURCE-STATEMENT LEVEL TO ASSIST IN PERFORMANCE EVALUATION AND DETERMINATION OF TEST CASE COVERAGE. (DAN 134)

ANIMATION
ANIMATION (IS AN ACTIVITY WHICH--ED.) DISPLAYS THE BEHAVIOR OF A MODEL IN TERMS OTHER THAN THOSE IN WHICH THE MODEL ITSELF IS TO BE EXPRESSED. IT IS USED TO DEMONSTRATE THE MODEL'S BEHAVIOR TO INTERESTED PARTIES UNWILLING, OR UNABLE, TO DEDUCE THIS FROM THE DESCRIPTION OF THE MODEL ITSELF, IN ORDER TO SEEK THEIR ACKNOWLEDGEMENT THAT THE MODEL CONFORMS TO SOME REQUIREMENT. ANIMATION CAN BE USED AS A TECHNIQUE TO COMMUNICATE WITH A CUSTOMER TO ENSURE THAT THE FORMAL SPECIFICATIONS SATISFY THE CUSTOMERS REQUIREMENTS. INTERACTION OF REQUIREMENTS REVISIONS, SPECIFICATION CHANCES AND RE-ANIMATION MUST CONTINUE UNTIL THE CUSTOMER SIGNS OFF. (DAN 874) SEE ALSO ANIMATION TOOLS.

ANIMATION TOOLS AND/OR TECHNIQUES
ANY PROCEDURE, DEVICE, OR TECHNIQUE WHICH CAN BE USED TO ANIMATE A MODEL. EXAMPLES ARE TEST TOOLS, THE TESTING PROCESS, PETRI NETS, META-PROGRAMMING, META-LANGUAGES, ETC. (DAN 874)

APOLLO FLIGHT SOFTWARE
SOFTWARE FOR THE APOLLO FLIGHT INCLUDED THE EXECUTIVE, DISPLAY, INTERFACE, INTERPRETER, MUCH OF THE HARDWARE INTERFACE LOGIC, INTERRUPT HANDLING AND COMPUTER SELF-TEST. (DAN 292)

APPLICATION-ORIENTED LANGUAGE
AN APPLICATION-ORIENTED LANGUAGE IS ONE WHICH HAS FACILITIES AND/OR NOTATIONS WHICH ARE USEFUL PRIMARILY FOR A SINGLE APPLICATION AREA. (E.G. A LANGUAGE FOR STATISTICAL ANALYSIS OR MACHINE DESIGN). (DAN 448)

APPLICATIONS SOFTWARE
SOFTWARE/PROGRAM SPECIFICALLY PRODUCED FOR A PARTICULAR USE OF THE COMPUTER SYSTEM. APPLICATION SOFTWARE/PROGRAMS USUALLY REQUIRE AN OPERATING SYSTEM FOR GENERAL CONTROL. USUALLY AN APPLICATION PROGRAM CONSISTS OF A NET OF INTEGRATED TASKS TO PROVIDE A PRIMARY SYSTEM FUNCTION SUCH AS NAVIGATION OR GUN FIRE CONTROL. (DAN 1201-MODIFIED)

ARBITER
A MECHANISM FOR EFFECTING THE MUTUALLY EXCLUSIVE USE OF A SHARED RESOURCE AMONG CONCURRENT PROCESSES. (DAN 1153)

ARCHITECTURAL DESIGN
SELECTION AMONG MAJOR ALTERNATIVES RELATIVE TO CONTROL LOGIC AND DATA STRUCTURAL TOPOLOGIES, MODULE COUPLING MODES, CLOCKING, PROTOCOLS, RESOURCE ALLOCATION STRATEGIES, ETC., TO THAT DEGREE OF DETAIL WHICH PROVIDES CONVINCING EVIDENCE OF PRODUCTION FEASIBILITY AND WHICH PERMITS COST AND SCHEDULE ESTIMATES OF PREDEFINED ACCURACY. (DAN 1153)

ARCHITECTURAL FAMILIES
A DOMAIN OF MACHINES BELONGS TO AN ARCHITECTURAL FAMILY IF THE MACHINES HAVE ONLY DIFFERENCES IN INSTRUCTION SET. SUCH MACHINES CAN,HOWEVER, HAVE SIGNIFICANT VARIATION IN OPERATING SYSTEM FUNCTIONS AND SERVICES, WHICH AFFECT PROGRAMS AND EXTERNAL INTERFACES.

ARTIFICIAL INTELLIGENCE
(1) THE CAPABILITY OF A DEVICE TO PERFORM FUNCTIONS THAT ARE NORMALLY ASSOCIATED WITH HUMAN INTELLIGENCE, SUCH AS REASONING, LEARNING, AND SELF-IMPROVEMENT. (ANSI-X3)

ASSEMBLE
TO TRANSLATE A SET OF SOME LANGUAGE STATEMENTS INTO A SIMPLE FORM, USUALLY THE MACHINE CODE OF A PARTICULAR MACHINE. THE TRANSLATION IS OFTEN A ONE-TO-ONE TRANSFORMATION. (ANSI-X3H1)

ASSEMBLER
A COMPUTER PROGRAM USED TO ASSEMBLE. SYNONYMOUS WITH ASSEMBLY PROGRAM. (ANSI-X3) (2) A TOOL THAT TRANSLATES PROGRAMS WRITTEN IN SYMBOLIC MACHINE LANGUAGE INTO ACTUAL MACHINE LANGUAGE PROGRAMS. (DAN LD7)

ASSEMBLY LANGUAGE
A LOW LEVEL PROGRAMMING LANGUAGE WHICH IS ACTUALLY A SYMBOLIC MACHINE LANGUAGE. (NASA)

ASSERTION

AN ASSERTION IS A LOGICAL EXPRESSION THAT SPECIFIES AN INSTANTANEOUS CONDITION OR RELATION AMONG THE VARIABLES OF A PROGRAM. ASSERTIONS ARE USED IN VARIOUS METHODS OF PROGRAM VERIFICATION AS WELL AS FOR PROGRAM TESTING, SYNTHESIS, AND ABSTRACTION. (SET) (2) A STATEMENT DEFINING PROPERTIES OR BEHAVIOR AT A SPECIFIC POINT IN A COMPUTER PROGRAM. (NASA)

ASSESSMENT OF CORRECTNESS
THE PROCESS OF JUDGING THAT A PROGRAM (OR PART) IS CORRECT, BASED ON A PARTIAL DEMONSTRATION OF ITS ACTUAL OR ENVISIONED BEHAVIOR. DEMONSTRATION MAY RANGE FROM RIGOROUS, FORMAL MATHEMATICAL PROOF TO INFORMAL RATIONALE, OR FROM EXHAUSTIVE TESTING TO MERE PROGRAM CHECKOUT. (DAN 1153)

ASSIGNMENT STATEMENT
AN INSTRUCTION USED TO EXPRESS A SEQUENCE OF OPERATIONS, OR USED TO ASSIGN OPERANDS TO SPECIFIED VARIABLES OR SYMBOLS, OR BOTH. (ANSI-X3) (2) ALL STATEMENTS THAT CHANGE THE VALUE OF A VARIABLE AS THEIR MAIN PURPOSE (E.G. ASSIGNMENT OR READ STATEMENTS, BUT THE ASSIGNMENT OF THE DO LOOP VARIABLE IN A DO STATEMENT SHOULD NOT BE INCLUDED). (SEL)

ASSURANCE TECHNOLOGY
THE BODY OF TECHNOLOGY USEFUL IN FOSTERING, ENSURING, AND CONFIRMING THAT A SOFTWARE PRODUCT PROPERLY FULFILLS ITS INTENDED PURPOSE.(S) (NASA)

ASTROS
A JOINT SPACE AND MISSILE TEST CENTER (SAMTEC) AND RADC EFFORT TO VALIDATE THE CLAIMED BENEFITS FROM THE APPLICATION OF MODERN PROGRAMMING PRACTICES IN AN AIR FORCE OPERATIONAL ENVIRONMENT. (DAN 226)

ATTACK
AN ATTACK ON A SYSTEM IS ANY DEFINED CIRCUMSTANCE WHICH RESULTS IN A GIVEN PROBABILITY OF ERROR, FAILURE, ERROR DETECTION, ERROR CORRECTION, SECURITY BREACH, ETC. AN ATTACK MIGHT ASSUME THE FORM OF SABOTAGE, INVALID DATA VALUES, INVALID COMBINATIONS OF VALID DATA ELEMENT VALUES IN INPUT, A PROGRAM LOGIC ERROR, ATTEMPT BY AN OPERATOR TO MOUNT AN OLD GENERATION OF THE "CORRECT" FILE, BREAKDOWN OF A COMPUTER'S AIR-CONDITIONING DURING A HEAT WAVE, ETC. (DAN 781)

ATTACK PROBABILITY
THE PROBABILITY OF AN ATTACK OF A GIVEN TYPE ON A GIVEN SYSTEM DURING A SPECIFIED TIME INTERVAL. (DAN 781)

ATTACK REPULSION PROBABILITY
SYNONOMOUS WITH SECURITY PROBABILITY. (DAN 781)

ATTITUDE/ORBIT
ANY SOFTWARE COMPONENT THAT IS DIRECTLY RELATED TO EITHER THE ATTITUDE DETERMINATION (OR CONTROL) TASK OR THE ORBIT DETERMINATION (OR CONTROL) TASK FALLS INTO THIS CATEGORY. THIS SHOULD INCLUDE FULL SYSTEMS IN GENERAL (SUCH AS GTDS, OR ISEE-B ATTITUDE) AS WELL AS SPECIFIC MODULES SUCH AS DETERMINISTIC ATTITUDE OR DCCONES. (SEL)

ATTRIBUTE LIST
A LIST OF THE IDENTIFIERS USED BY A PROGRAM DESCRIBING THE CHARACTERISTICS OF THOSE IDENTIFIERS, AND SHOWING THE SOURCE STATEMENTS WHERE THEY ARE FIRST

DEFINED (OR FIRST USED), AND, FOR VARIABLES, THEIR (RELATIVE) STORAGE LOCATIONS. (SEL)

AUDIT

A SOFTWARE AUDIT IS A REVIEW BY OUTSIDE (NOT INVOLVED IN THE DEVELOPMENT OF THE PROJECT) SOFTWARE DEVELOPMENT EXPERTS FOR THE PURPOSES OF ASSESSING PROGRESS, MAINTAINING SCHEDULES, PRODUCING RECOMMENDATIONS CONCERNING AREAS OR CONCEPTS THAT HAVE BEEN OVERLOOKED AND RATING THE RELATIVE EFFICIENCIES OF VARIOUS APPROACHES TO PROBLEMS. (DAN 300) (2) A FORMAL OR OFFICIAL EXAMINATION THAT ATTESTS TO THE CONFORMITY (OR NON-CONFORMITY) BETWEEN TWO SUPPOSED EQUIVALENT ENTITIES, ACCORDING TO A PREDEFINED SET OF RULES. (DAN 1153) (3) THE FOLLOWING OF OPERATING SYSTEM TRACERS RECORDED TO DOCUMENT ACCOUNTABILITY IN COST, EQUIPMENT USED, AND FILES ACCESSED.

AUGMENTABILITY

CODE POSSESSES THE CHARACTERISTIC AUGMENTABILITY TO THE EXTENT THAT IT CAN EASILY ACCOMODATE EXPANSION IN COMPONENT COMPUTATIONAL FUNCTIONS OR DATA STORAGE REQUIREMENTS. THIS IS A NECESSARY CHARACTERISTIC FOR MODIFIABILITY. (DAN 239)

AUTOMATA THEORETIC

A TESTING STRATEGY WHICH HAS THE FOLLOWING CHARACTERISTICS: A) ONLY THE CONTROL STRUCTURE OF THE DESIGN IS CHECKED. B) IT DOES NOT REQUIRE AN "EXECUTABLE" SPECIFICATION. C) TEST SEQUENCES ARE GUARANTEED TO REVEAL ANY ERRORS IN THE CONTROL STRUCTURE, PROVIDED THAT SOME REASONABLE ASSUMPTIONS ARE SATISFIED. (DAN 308)

AUTOMATABILITY

"AUTOMATABLE" MEANS THAT THE HUMAN OPERATOR NOT ONLY CAN CONTROL THE SYSTEM COMPLETELY MANUALLY BUT ALSO CAN DEFINE PORTIONS OF THE SYSTEM OPERATIONS AS PROCEDURES TO BE PERFORMED AUTOMATICALLY BY THE SYSTEM. (DAN 346) AUTOMATABILITY WOULD THUS INDICATE THE DEGREE TO WHICH A SYSTEM IS AUTOMATABLE. (ED)

AUTOMATE

TO CONVERT A PROCESS/PROCEDURE DONE MANUALLY TO A PROCESS/ PROCEDURE DONE AUTOMATICALLY.

AUTOMATED DESIGN TOOLS

COMPUTER PROGRAMS USED TO PROVIDE AN UNDERSTANDABLE REPRESENTATION OF THE SOFTWARE DESIGN AS IT EVOLVES.

AUTOMATED DOCUMENTATION

DOCUMENTATION WHICH IS PRODUCED BY AUTOMATED MEANS, USUALLY BY A SPECIALIZED PROGRAM OR A PROGRAM LIBRARY SYSTEM.

AUTOMATED ERROR DETECTION

THE USE OF AUTOMATED MEANS TO DETECT INCONSISTENCIES BETWEEN ASSERTIONS ABOUT THE INPUTS AND OUTPUTS OF THE VARIOUS ELEMENTS OF THE SOFTWARE

AUTOMATED PATH ANALYSIS

A SOFTWARE TECHNIQUE WHICH SCANS SOURCE CODE IN ORDER TO DESIGN AN OPTIONAL SET OF TEST CASES TO EXERCISE THE PRIMARY PATHS IN A SOFTWARE MODULE. (DAN 142)

AUTOMATED PROGRAM PROVING
SEE AUTOMATED VERIFICATION TOOLS.

AUTOMATED TEST GENERATOR
A COMPUTER PROGRAM THAT ACCEPTS INPUTS SPECIFYING A TEST SCENARIO IN SOME SPECIAL LANGUAGE, GENERATES THE EXACT COMPUTER INPUTS, AND DETERMINES THE EXPECTED RESULTS. (DAN 134)

AUTOMATED TESTING
TESTING, USUALLY BY A SOFTWARE PROGRAM WHICH IS GENERATED BY ALGORITHMS AND WHICH CONSTITUTES AN EFFECTIVE TEST FOR A SOFTWARE SYSTEM OR A COMPONENT OF THE SYSTEM. (DAN 234)

AUTOMATED TOOLS
ANY PROGRAMS WHOSE PURPOSE IS TO AID IN SOFTWARE DEVELOPMENT (E.G., COMPILER, TEXT EDITOR, DUMP OR TRACE FACILITY, ETC.). THIS INCLUDES COMPILERS BUT NOT STANDARD OPERATING SYSTEM SOFTWARE (E.G., LINK EDITOR). (SEL) (2) COMPUTER PROGRAMS WHICH PERFORM VARIOUS SOFTWARE DESIGN, ANALYSIS, TEST, AND MAINTENANCE FUNCTIONS THROUGH THE AUTOMATION OF ASSOCIATED METHODS OR PROCEDURES. (NASA)

AUTOMATED UNIT TEST (AUT)
A MODULE TEST DRIVER TOOL DEVELOPED FOR USE WITHIN IBM CORP.

AUTOMATED VERIFICATION SYSTEMS
COMPUTER PROGRAMS THAT INSTRUMENT THE SOURCE CODE BY GENERATING AND INSERTING COUNTERS AT STRATEGIC POINTS TO PROVIDE MEASURES OF TEST EFFECTIVENESS. THEY PROVIDE DATA THAT DETAILS HOW THOROUGHLY THE SOURCE CODE HAS BEEN EXERCISED. (DAN 134)

AUTOMATED VERIFICATION TOOLS
AUTOMATED TOOLS FOR QUANTIFYING THE EFFECTIVENESS OF TEST DATA IN TERMS OF EXERCISING THE PROGRAM CONTROL STRUCTURES. SOME AUTOMATED VERIFICATION TOOLS CAN BE USED TO GENERATE DESCRIPTIVE PROGRAM DOCUMENTATION REPORTS, PROVIDE DYNAMIC EXECUTION TRACES OF MODULES AND DECISION-TO-DECISION (DD) PATHS, ASSIST IN GENERATION OF ADDITIONAL TEST CASES AND FLAG UNEXPECTED EXECUTION BEHAVIOR THROUGH THE USE OF COMPUTATION DIRECTIVES. (DAN 393)

AUTOMATIC DATA COLLECTION
THE COLLECTION OF DATA ABOUT A PROGRAM BY AUTOMATED MEANS, USUALLY DURING THE EXECUTION OF THE PROGRAM. THE COLLECTED DATA/STATISTICS MAY BE PRINTED OUT AT THE END OF A PROGRAM'S EXECUTION OR MAY BE STORED AUTOMATICALLY. THE DATA/STATISTICS MAY INCLUDE STATEMENT FREQUENCY PROFILES, DYNAMIC STATEMENT COUNTS, POST-MORTEM DUMPS, TRACE TABLES, AND OTHER FORMS OF COLLECTED DATA. (DAN 437-MODIFIED)

AUTOMATIC DEBUGGING
SYNONOMOUS WITH AUTOMATED ERROR DETECTION

AUTOMATIC PROGRAMMING
THE PROCESS OF USING A COMPUTER TO PERFORM SOME STAGES OF THE WORK INVOLVED IN PREPARING A COMPUTER PROGRAM. SYNONOMOUS WITH AUTOMATIC CODING. (ANSI-X3) (2) USE OF MACHINE INTERACTIVE TECHNIQUES TO SELECT PROGRAM MODULES ALREADY IN A PROGRAM LIBRARY TO BE USED AS MODULES IN A NEW PROGRAM OR PROJECT. (DAN

232)

AUTOMATIC SOFTWARE TEST DRIVERS
A SOFTWARE TOOL WHICH CONTROLS AND MONITORS THE EXECUTION OF SOFTWARE TESTS. (DAN 282)

AVAILABILITY
AVAILABILITY IS THE PROBABILITY THAT COMPUTER SOFTWARE IS "UP" OR CAPABLE OF FUNCTIONING IN ACCORDANCE WITH REQUIREMENTS AT ANY TIME. THIS PROBABILITY IS OFTEN MEASURED AS THE RATIO OF "UP" TIME TO TOTAL NEED TIME... THE COMPUTER SOFTWARE MAY BE CLASSIFIED AS NOT AVAILABLE IF IT IS BLOCKED BY ANOTHER USER, OR IF IT CONTAINS ERRORS AND IS BEING CORRECTED. (SET) (2) THE RATIO OF SYSTEM UP-TIME TO THE TOTAL OPERATING TIME. (DAN 232) (3) THE PROBABILITY THAT A SYSTEM IS OPERATING SATISFACTORILY AT ANY POINT IN TIME, WHEN USED UNDER STATED CONDITIONS. (DAN 781) (4) THE PROBABILITY THAT A SYSTEM, SUBSYSTEM, OR COMPONENT WILL BE FUNCTIONALLY READY OR OPERABLE AT SOME SPECIFIED POINT IN TIME. (NASA)

AVAILABILITY MODEL
A MODEL (OR MODELS) WHICH PREDICTS THE EXPECTED RATIO OF SYSTEM UP-TIME TO THE TOTAL OPERATING TIME. (DAN 232)

AVIONICS APPLICATIONS
SOFTWARE ENGINEERING APPLIED TO FLIGHT CONTROL SYSTEMS FOR AIRCRAFT.(DAN 258)

BALLISTIC MISSILE DEFENSE
INDEXING TERM. REFERS TO SOFTWARE USED AS A COMPONENT IN BALLISTIC MISSILE DEFENSE SYSTEMS.

BASELINE DIAGRAM
AN ORDERED CHART LISTING ALL COMPONENTS IN A SYSTEM WHERE A CONNECTION FROM A HIGHER COMPONENT TO A LOWER ONE INDICATES THAT THE HIGHER COMPONENT CALLS THE LOWER ONE. (SEL)

BASELINE PROGRAM
A PROGRAM POSSESSING WELL DEFINED CAPABILITIES AND FUNCTIONS WHICH IS DECREED TO BE THE STARTING POINT FOR FURTHER PROGRAM DEVELOPMENT. (DAN 1201)

BASIC
BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE. AN ALGEBRAIC PROBLEM-ORIENTED HIGH LEVEL PROGRAMMING LANGUAGE INTENDED FOR INTERACTIVE USE.

BATCH PROCESSING
THE PROCESSING OF DATA OR THE ACCOMPLISHMENT OF JOBS ACCUMULATED IN ADVANCE IN SUCH A MANNER THAT EACH ACCUMULATION THUS FORMED IS PROCESSED OR ACCOMPLISHED IN THE SAME RUN. (ANSI-X3) (2)PERTAINING TO THE TECHNIQUE OF EXECUTING A SET OF COMPUTER PROGRAMS SUCH THAT EACH IS COMPLETED BEFORE THE NEXT PROGRAM OF THE SET IS STARTED. (ANSI-X3) (3) USAGE OF A COMPUTER WHERE THE ENTIRE JOB IS READ INTO THE MACHINE BEFORE THE PROCESSING BEGINS. (INTERACTIVE USAGE ALWAYS IS VIA A TERMINAL, BATCH USAGE MAY BE VIA A TERMINAL OR A CARD DECK.) (SEL)

BAYESIAN MODEL
INDEXING TERM. REFERS TO THE MATHEMATICAL METHODOLOGY USED TO CONSTRUCT, OR WHICH IS THE FORM ASSUMED BY, A PARTICULAR MODEL.

BEBUGGING
SYNONOMOUS WITH BUG SEEDING/TAGGING

BEGIN-END BLOCK
BEGIN-END BLOCK IS A COLLECTION OF COMPUTER PROGRAM STATEMENTS BRACKETED BY BEGIN AND END STATEMENTS. THE LATTER DELIMITS THE SCOPE OF NAMES AND IS ALSO ACTIVATED BY NORMAL SEQUENTIAL FLOW OF CONTROL. ...THE TERM CAME FROM THE ALGOL 60 PROGRAMMING LANGUAGE. THESE STATEMENTS ARE OFTEN USED TO DEFINE THE LIMITS OF A COLLECTION OF CODE SUCH AS A MODULE OR SUBROUTINE. (SET)

BEHAVIOR MODELLING
DESCRIBING WHAT A COMPONENT OF A SOFTWARE SYSTEM WILL DO IN TERMS OF AN ABSTRACTION OF THE COMPONENT'S OPERATION WHICH FOCUSES UPON EFFECT RATHER THAN CAUSE. (DAN 242)

BEHAVIORAL MODEL
A MATHEMATICAL FUNCTION THAT RELATES CAUSE AND EFFECT QUANTITATIVELY. (DAN 255)

BIT
A CONTRACTION OF THE TERM "BINARY DIGIT" AND HENCE EITHER A 0 OR A 1 IN THE BASE-TWO NUMBER SYSTEM. (NASA)

BLACK BOX
AN ACTUAL OR A CONCEPTUAL DEVICE WHICH TRANSFORMS INPUT DATA INTO OUTPUT DATA ACCORDING TO A PRESCRIBED FUNCTIONAL RELATIONSHIP, BUT WHOSE INTERNAL MECHANIZATION IS NOT NECESSARILY KNOWN. (NASA)

BLOCK DIAGRAM
A DIAGRAM OF A SYSTEM, INSTRUMENT, OR COMPUTER, IN WHICH THE PRINCIPAL PARTS ARE REPRESENTED BY SUITABLY ASSOCIATED GEOMETRICAL FIGURES TO SHOW BOTH THE BASIC FUNCTIONS AND THE FUNCTIONAL RELATIONSHIPS AMONG THE PARTS. (ANSI-X3)

BLOCK-STRUCTURED LANGUAGE
A HIGHER-ORDER PROGRAMMING LANGUAGE WHICH DEMARCATES RELATED SEQUENCES OF CODE, OR BLOCKS, USUALLY WITH THE STATEMENTS BEGIN AND END. (NASA)

BOTTOM-UP DESIGN
THE DESIGN OF THE SYSTEM STARTING WITH THE LOWEST LEVEL ROUTINES AND PROCEEDING TO THE HIGHER LEVEL ROUTINES THAT USE THE LOWER LEVELS. (SEL) CONTRAST WITH TOP-DOWN DESIGN.

BOTTOM-UP IMPLEMENTATION
THE IMPLEMENTATION OF THE SYSTEM STARTING WITH THE LOWEST LEVEL ROUTINES AND PROCEEDING TO THE HIGHER LEVEL ROUTINES THAT USE THE LOWER LEVELS. (SEL) CONTRAST WITH TOP-DOWN IMPLEMENTATION.

BUDGETING AND ESTIMATING
THOSE ACTIVITIES THAT DETERMINE THE LEVELS OF EFFORT AND RESOURCES NEEDED TO ACCOMPLISH A PROJECT. (DAN LD-7)

BUG
ONE OR MORE SOFTWARE BUGS EXIST IN A SYSTEM IF A SOFTWARE CHANGE IS REQUIRED TO CORRECT A SINGLE MAJOR ERROR OR MINOR ERROR SO AS TO MEET SPECIFIED OR IMPLIED SYSTEM PERFORMANCE REQUIREMENTS. (DAN 31)

BUG SEEDING/TAGGING
THE PROCESS OF ADDING BUGS (OR ERRORS) TO THOSE ALREADY ASSUMED TO BE IN A PROGRAM WITH THE PURPOSE OF OBTAINING AN ESTIMATE FOR THE NUMBER OF NATURAL BUGS REMAINING IN THE PROGRAM. IT IS ALSO ASSUMED THAT RATIO OF THE NUMBER OF UNDISCOVERED SEEDED BUGS TO THE TOTAL NUMBER OF BUGS SEEDED CAN SERVE AS AN INDICATION OF THE DEGREE OF "DEBUGGEDNESS" OR RELIABILITY OF THE PROGRAM. (DANS 232 AND 781)

BUILDING BLOCK
GENERATION OF A PROGRAM AS AN ISOLATED BUILDING BLOCK. NECESSARY INDEPENDENT SUBPROGRAMS ARE GENERATED FIRST, FOLLOWED BY GENERATION OF THE DEPENDENT FUNCTIONS. (DAN 1201)

BUILDS
BUILDS ARE FUNCTIONALLY-ORIENTED SECTIONS OF A MORE COMPLEX SOFTWARE DEVELOPMENT PROJECT. THE "BUILDS" APPROACH TO SOFTWARE DEVELOPMENT IS DESIGNED TO IMPROVE THE QUALITY OF THE TESTING PROCESS BY MAINTAINING A VISIBLE CONNECTION BETWEEN REQUIREMENTS AND THE TEST PLANS AND PROCEDURES DURING THE ENTIRE DEVELOPMENT PROCESS. (DAN 326)

BUILT-IN FLEXIBILITY
BUILT-IN FLEXIBILITY IS THE ABILITY OF A SYSTEM TO IMMEDIATELY HANDLE DIFFERENT LOGICAL SITUATIONS. BUILT-IN FLEXIBILITY INCREASES SYSTEM COMPLEXITY PROPORTIONATELY. IN A WELL-DESIGNED SYSTEM THE INITIAL MEASURE OF BUILT-IN FLEXIBILITY WILL BE ALMOST EQUAL TO THE COMPLEXITY MEASURE. (DAN 781)

BUILT-IN-TEST
TEST CAPABILITY WHICH IS INTEGRAL TO A UNIT AND WHICH MAY PERFORM SYSTEM CHECKS AS WELL AS SELF-TEST FUNCTIONS. (NASA)

BUSINESS AND FINANCIAL APPLICATIONS
SOFTWARE OR SOFTWARE SYSTEM COMPONENTS RELATED TO SOME ACCOUNTING TASK, FINANCIAL DATA FORMATTING, BUSINESS DATA RETRIEVAL OR REPORTING, OR POSSIBLY PERSONNEL DATA MANAGEMENT. (SEL)

BYTE
A STRING OF BITS WHOSE LENGTH IS THE SMALLEST ACCESSIBLE AS A UNIT IN A COMPUTER MEMORY; ALSO, THE LENGTH USED TO REPRESENT A CHARACTER. (NASA)

CALIBRATION ERROR
AN ERROR PURPOSELY INSERTED INTO A PROGRAM TO SERVE AS A MEANS FOR GAUGING THE COMPLETENESS OF TESTING TO UNCOVER INDIGENOUS ERRORS. (DAN 1153)

CAPABILITY
A CAPABILITY IS DEFINED AS AN ABSTRACT ENCAPSULATION OF THE DATA NEEDED TO DEFINE ACCESS TO A PROTECTED OBJECT. --WITH RESPECT TO SECURITY. (DAN 724) (2) CAPABILITIES ARE DISCRETELY IDENTIFIED ELEMENTS OF PERFORMANCE WHICH ARE EXPECTED (EITHER FORMALLY OR INFORMALLY) OF A PRODUCT OR COMBINATION OF

PRODUCTS. A FAILURE IS THE ABSENCE OF ONE OR MORE CAPABILITIES DURING THE USE OF A PRODUCT. SEVERITY OF A FAILURE IS DIRECTLY PROPORTIONAL TO THE VALUE OF THE ABSENT CAPABILITIES TO THE USER. AN ERROR BECOMES A FAILURE WHEN SOFTWARE IS INCAPABLE OF RE-ESTABLISHING ITS CAPABILITIES IN AN ERROR ENVIRONMENT. --WITH RESPECT TO EFFECTIVENESS. (DAN 749)

CAPABILITY MACHINE
A SET OF HARDWARE-SOFTWARE MECHANISMS USED TO IMPLEMENT SECURE OR FAULT-TOLERANT COMPUTING SYSTEMS WHICH MAY INCLUDE WELL-DEFINED RIGHTS TO ACCESS CERTAIN RESOURCES AT VARIOUS LEVELS AND VALIDATION KEYS. THE MECHANISMS MAY ALSO BE REFERRED TO AS CAPABILITY MONITORS OR CAPABILITY MANAGERS.

CASE
A CASE STATEMENT IS A STATEMENT THAT TRANSFERS CONTROL TO ONE OF SEVERAL LOCATIONS DEPENDING ON THE VALUE OF THE CONTROL EXPRESSION. ...THE "CASE" CONSTRUCT PROVIDES A N-WAY TRANSFER OF CONTROL AND IS CONSIDERED A "GOTO" REPLACEMENT. ONE TYPE OF CASE STATEMENT IS THE "ARITHMATIC IF" IN FORTRAN. (SET)

CERTIFICATION
CERTIFICATION EXTENDS THE PROCESSES OF VERIFICATION AND VALIDATION TO AN OPERATIONAL ENVIRONMENT; CONFIRMS THAT THE SYSTEM IS OPERATIONALLY EFFECTIVE, IS CAPABLE OF SATISFYING REQUIREMENTS UNDER SPECIFIED OPERATING CONDITIONS; AND FINALLY GUARANTEES ITS COMPLIANCE WITH REQUIREMENTS IN WRITING. CERTIFICATION USUALLY IMPLIES THE EXISTENCE OF AN INDEPENDENT QUALITY CONTROL GROUP FOR THE ACCEPTANCE TESTING OF THE OVERALL SYSTEM. THE ACCEPTANCE TESTING MAY BE ACCOMPLISHED BY OPERATIONAL TESTING, LABORATORY TESTING, AND/OR PLACING THE SYSTEM IN SIMULATED OPERATION. (SET) (2) THE FORMAL DEMONSTRATION OF SYSTEM ACCEPTABILITY TO OBTAIN AUTHORIZATION FOR ITS OPERATIONAL USE. (NASA)

CERTIFICATION PLAN
AN APPROVED DOCUMENT CONTAINING A PLAN TO PROVIDE SOFTWARE CERTIFICATIONS. (DAN 1201)

CERTIFICATION TEST
THE FORMAL DEMONSTRATION TO THE CUSTOMER OF THE TESTS DOCUMENTED IN THE CERTIFICATION TEST PROCEDURE. (DAN 1201)

CERTIFICATION TEST PROCEDURES
A FORMAL DOCUMENT DETAILING THE ACTIONS TO BE PERFORMED AND RESULTS TO BE OBSERVED IN VERIFYING THE CORRECT OPERATION OF A PROGRAM. (DAN 1201)

CERTIFICATION TOOLS
INFORMATION REPORTING AND SUMMARIZING COMPONENTS THAT PROVIDE A SYSTEMATIC DATA COLLECTION AND SUMMARIZING MECHANISM THAT PRODUCES A SYSTEM STATUS REPORT. (DAN LD-7)

CHANGE
A MODIFICATION TO DESIGN, CODE, OR DOCUMENTATION. A CHANGE MIGHT BE MADE TO CORRECT AN ERROR, TO IMPROVE SYSTEM PERFORMANCE, TO ADD A CAPABILITY, TO IMPROVE APPEARANCE, TO IMPLEMENT A REQUIREMENTS CHANGE, ETC., (SEL) (2) ANY ALTERATION (ADDITION, DELETION, CORRECTION) OF THE PROGRAM CODE WHETHER IT

BE A SINGLE CHARACTER OR THOUSANDS OF LINES OF CODE. CHANGES MADE TO IMPROVE DOCUMENTATION OR SATISFY NEW SPECIFICATIONS ARE IMPORTANT TO RECORD AND STUDY, BUT ARE NOT COUNTED AS BUGS. (DAN LD-7) --COMPARE WITH MAINTENANCE OR WITH MODIFICATION

CHARACTER CODE
A CORRESPONDENCE BETWEEN A CHARACTER SET AND A SET OF INTEGERS. (ANSI-X3-H1)

CHARACTER SET
A SET OF GRAPHIC SYMBOLS INDEPENDENT OF FONT. A CHARACTER SET DOES NOT INCLUDE SPECIFICATION OF CODES TO REPRESENT CHARACTERS. (ANSI-X3H1)

CHEBYSHEV'S INEQUALITY
INDEXING TERM. REFERS TO THE MATHEMATICAL METHODOLOGY USED TO CONSTRUCT, OR WHICH IS THE FORM ASSUMED BY, A PARTICULAR MODEL.

CHIEF PROGRAMMER TEAM
A CHIEF PROGRAMMER TEAM IS A STRUCTURED TEAM OF SPECIALISTS FOR SOFTWARE DEVELOPMENT HEADED BY A CHIEF PROGRAMMER. SPECIALIZED ROLES FOR PEOPLE ON A PROJECT AND THE RELATIONSHIPS AMONG THEM ARE WELL DEFINED. THE CHIEF PROGRAMMER TEAM HAS AS ITS CORE THREE MEMBERS: THE CHIEF PROGRAMMER, THE BACKUP PROGRAMMER, AND THE SECRETARY/LIBRARIAN. THE THREE PERSONS PERFORM DIFFERENT FACETS OF THE ONE JOB, SOFTWARE DEVELOPMENT. THEY FUNCTION IN CONCERT IN JOBS THAT SUPPORT AND COMPLEMENT EACH OTHER. THE CHIEF PROGRAMMER IS A SENIOR LEVEL PROGRAMMER WHO IS RESPONSIBLE FOR THE DETAILED DEVELOPMENT OF THE SOFTWARE ASSIGNED TO THAT TEAM. HE PRODUCES A CRITICAL NUCLEUS OF THE SYSTEM IN FULL AND SPECIFIES AND INTEGRATES ALL OTHER PROGRAMMING FOR THE SYSTEM AS WELL. COORDINATION IS THE PROVINCE OF THE CHIEF PROGRAMMER ALONE, THUS REDUCING THE NUMBER OF MINDS INVOLVED IN PROJECT COMMUNICATION BY A FACTOR OF FIVE OR SIX. THE BACKUP PROGRAMMER IS FAMILIAR WITH ALL ASPECTS OF THE SYSTEM AND CAN SUBSTITUTE FOR THE CHIEF PROGRAMMER WHEN NECESSARY. THE BACKUP PROGRAMMER CONTRIBUTES SIGNIFICANT PORTIONS OF THE PROGRAMMING EFFORT AND, ALONG WITH THE CHIEF PROGRAMMER, READS AND CRITIQUES THE CODE OF OTHER TEAM MEMBERS. THE SECRETARY/LIBRARIAN MAINTAINS THE STATUS OF PROGRAM AND TEST DATA IN SUCH A FORM THAT PROGRAMMERS CAN WORK MORE EFFECTIVELY. ALL DATA ENTRY, SOURCE AND TEST DATA UPDATING, COMPILATIONS AND TEST RUNS, AND DOCUMENTATION COORDINATION ARE PERFORMED BY THE SECRETARY/LIBRARIAN. THE REST OF THE TEAM CONSISTS OF PROGRAMMERS AS REQUIRED. THE TEAM IS NORMALLY LIMITED TO LESS THAN 10 MEMBERS. (SET) (2) THIS CONCEPT IMPLIES THE USE OF A HIGHLY STRUCTURED TEAM OF SPECIALISTS FOR SOFTWARE PRODUCTION RELYING ON TECHNICAL PROCEDURES WHICH ARE BASED ON STRUCTURED PROGRAMMING PRINCIPLES, AND OFFICE PROCEDURES BACKED UP WITH AUTOMATED AIDS TO SUPPORT GROUP COMMUNICATION. SPECIALIZED ROLES FOR PEOPLE ON THE TEAM, AND THE RELATIONSHIPS AMONG THEM, ARE WELL-DEFINED. (DAN 172)

CLARITY
CODE POSSESSES THE CHARACTERISTIC CLARITY TO THE EXTENT THAT IT IS CONCISE, STRAIGHTFORWARD (LACK OF TRICKY, OBSCURE CODE), UNDERSTANDABLE, HAS CLEAR CONTROL STRUCTURE, UNIFORM STYLE, SELF-CONTAINED WITH RESPECT TO DOCUMENTATION, MAKES APPROPRIATE USE OF MACROS AND OF CHANGE LEVELS. (DAN 748)

CLERICAL
THE PROCESS OF COPYING AN ITEM FROM ONE FORMAT TO ANOTHER, OR FROM ONE

MEDIUM TO ANOTHER, INVOLVING NO INTERPRETATION OR SEMANTIC TRANSLATION. (SEL)

COBOL
(COMMON BUSINESS ORIENTED LANGUAGE). A PROGRAMMING LANGUAGE DESIGNED FOR BUSINESS DATA PROCESSING. (ANSI-X3)

CODE
THE SYMBOLIC REPRESENTATION OF COMPUTER PROGRAM STATEMENTS. (NASA)

CODE AUDITING
CODE AUDITING IS THE PROCESS OF VERIFYING ADHERENCE TO PROGRAMMING STANDARDS.

CODE INSPECTION
CODE ANALYSIS IS THE PROCESS OF VERIFYING THAT THE COMPUTER PROGRAM, AS CODED, IS A CORRECT IMPLEMENTATION OF THE SPECIFIED DESIGN. CODE READING IS THE VISUAL INSPECTION OF THE SOURCE CODE BY PERSONS OTHER THAN THE CREATOR OF THE CODE. (SEL) --COMPARE WITH CODE VERIFICATION.

CODE STANDARDS AUDITOR
A COMPUTER PROGRAM USED TO AUTOMATICALLY DETERMINE WHETHER PRESCRIBED PROGRAMMING STANDARDS AND PRACTICES HAVE BEEN FOLLOWED.

CODE VERIFICATION
THE PROCESS OF DETERMINING WHETHER THE ACTUAL CODE IS COMPLIANT WITH THE TECHNICAL DESCRIPTION OF THE COMPUTER PROGRAM SPECIFICATION. THE ANALYSIS PERFORMED IS VERY DETAILED AND SEEKS TO IDENTIFY ERRORS OR DISCREPANCIES THAT STEM FROM INCONSISTENT USE OF INSTRUCTIONS, INCORRECT LOGIC FLOW, INCOMPATIBLE INTERFACES, FAILURES TO MEET TIMING AND SIZING BUDGETS, AND/OR INACCURACIES IN SIZING OR CALCULATIONS. (SET)

CODER
AN INDIVIDUAL MAINLY INVOLVED WITH WRITING BUT NOT DESIGNING A COMPUTER PROGRAM. (DAN 1153)

CODING
THE GENERATION OF A SEQUENCE OF PRECISE STATEMENTS IN A FORM APPROPRIATE TO PERMIT A COMPUTER TO PERFORM AN INTENDED FUNCTION. (NASA) (2) THE ACTIVITY OF EXPRESSING THE STEPS OF A GIVEN ALGORITHM IN A COMPUTER LANGUAGE (OR, PERHAPS, MORE THAN ONE LANGUAGE). A UNIT IS NOT QUALIFIED AS "CODED" UNTIL COMPILED (OR ASSEMBLED) AND ALL SYNTAX ERRORS REMOVED. (DAN 1153)

COHESION OR MODULE STRENGTH
A RELATIVE MEASURE OF THE STRENGTH OF RELATIONSHIPS AMONG THE INTERNAL COMPONENTS OF A MODULE INSOFAR AS THEY CONTRIBUTE TO THE VARIATION IN ASSUMPTIONS MADE BY THE OUTSIDE PROGRAM CONCERNING THE ROLE THE MODULE PLAYS IN THE PROGRAM. INVARIANT ASSUMPTIONS ABOUT A MODULE INDICATE HIGH STRENGTH. (DAN 1153)

COINCIDENTAL CORRECTNESS
COINCIDENTAL CORRECTNESS OCCURS WHEN A SPECIFIC TEST POINT FOLLOWS AN INCORRECT PATH, AND YET THE OUTPUT VARIABLES COINCIDENTLY ARE THE SAME AS IF THAT TEST POINT WERE TO FOLLOW THE CORRECT PATH. (DAN 842)

COMMAND
TO DIRECT OR TO ISSUE AN ORDER. A COMMAND IS AN ORDER OR DIRECTION. (ANSI-X3H1)

COMMAND LANGUAGE
A SOURCE LANGUAGE CONSISTING PRIMARILY OF PROCEDURAL OPERATORS, EACH CAPABLE OF INVOKING A FUNCTION TO BE EXECUTED. (2) THE LANGUAGE THROUGH WHICH A USER DIRECTS A SYSTEM. (ANSI-X3H1)

COMMAND LEVEL
A MODE IN WHICH INPUT STATEMENTS ARE ACCEPTED BY A COMMAND PROCESSOR. (ANSI-X3H1)

COMMAND STATEMENT
A STATEMENT IN A COMMAND STATEMENT. (ANSI-X3H1)

COMMAND/CONTROL APPLICATIONS
SOFTWARE APPLICATIONS USED TO EITHER GENERATE VEHICLE COMMANDS OR TRANSMIT THESE COMMANDS FROM THE CONTROL CENTER. (SEL)

COMMENT
A STATEMENT OR PARTIAL STATEMENT INCLUDED WITHIN A SET OF COMMAND STATEMENTS WHICH IS NOT INTENDED FOR ANY PROCESSING BY AN OSCRL PROCESSOR OTHER THAN POSSIBLE OUTPUT. (ANSI-X3H1)

COMMUNICATIONS SWITCHING SYSTEM
INDEXING TERM. REFERS TO THE SOFTWARE COMPONENT OF A COMMUNICATIONS SWITCHING SYSTEM OR TO THE USE OF SOFTWARE AS A TOOL IN THE DEVELOPMENT OF A COMMUNICATIONS SWITCHING SYSTEM.

COMMUNICATIVENESS
CODE POSSESSES THE CHARACTERISTIC COMMUNICATIVENESS TO THE EXTENT THAT IT FACILITATES THE SPECIFICATION OF INPUTS AND PROVIDES OUTPUTS WHOSE FORM AND CONTENT ARE EASY TO ASSIMILATE AND USEFUL. COMMUNICATIVENESS IS NEEDED FOR TESTABILITY AND HUMAN ENGINEERING. (DAN 239)

COMPARATOR
A COMPUTER PROGRAM USED TO COMPARE TWO VERSIONS OF THE SAME COMPUTER PROGRAM UNDER TEST TO ESTABLISH IDENTICAL CONFIGURATION OR TO SPECIFICALLY IDENTIFY CHANGES IN THE SOURCE CODE BETWEEN THE TWO VERSIONS. (DAN 134)

COMPATIBILITY
COMPATIBILITY IS THE MEASURE OF PORTABILITY THAT CAN BE EXPECTED OF SYSTEMS WHEN THEY ARE MOVED FROM ONE GIVEN ENVIRONMENT TO ANOTHER. BY WAY OF CONTRAST, PORTABILITY IS A CHARACTERISTIC OF THE SYSTEM; COMPATIBILITY IS A RELATIONSHIP BETWEEN TWO ENVIRONMENTS. (DAN 781)

COMPETING CHARACTERISTICS
A SET OF FACTORS THAT RELATE TO THE FINAL QUALITY OF A PIECE OF SOFTWARE, BUT THAT MAY CONFLICT OR COMPETE FOR PROJECT OR MACHINE RESOURCES. THESE MAY BE ORDERED IN PRIORITY TO FORM IMPLEMENTATION GUIDELINES. (DAN 1153)

COMPILE
TO TRANSLATE A COMPUTER PROGRAM EXPRESSED IN A PROBLEM-ORIENTED LANGUAGE

INTO A COMPUTER-ORIENTED LANGUAGE. (ANSI-X3) (2) TO PREPARE A MACHINE LANGUAGE PROGRAM FROM A COMPUTER PROGRAM WRITTEN IN ANOTHER PROGRAMMING LANGUAGE BY MAKING USE OF THE OVERALL LOGIC STRUCTURE OF THE PROGRAM, OR GENERATING MORE THAN ONE COMPUTER INSTRUCTION FOR EACH SYMBOLIC STATEMENT, OR BOTH, AS WELL AS PERFORMING THE FUNCTION OF AN ASSEMBLER. (ANSI-X3) (3) TO TRANSLATE A SET OF SOURCE LANGUAGE STATEMENTS INTO A SIMPLE FORM, USUALLY THE ASSEMBLY CODE OR MACHINE CODE OF A PARTICULAR MACHINE. THE TRANSLATION IS OFTEN A ONE-TO-MANY TRANSFORMATION. (ANSI-X3H1)

COMPILER

A COMPUTER PROGRAM USED TO COMPILE. SYNONOMOUS WITH COMPILING PROGRAM. (ANSI-X3) (2) A TOOL, USED IN THE PRODUCTION OF SOFTWARE SYSTEMS, THAT ALLOWS PROGRAMS TO BE WRITTEN IN HIGHER-ORDER LANGUAGES. EXAMPLES INCLUDE THE PL/I COMPILER, FORTRAN COMPILER, AND COBOL COMPILER. (DAN LD7) (3) A PROGRAM WHICH TRANSLATES A HIGHER-ORDER LANGUAGE SOURCE PROGRAM INTO EITHER ASSEMBLY OR MACHINE LANGUAGE. (NASA)

COMPILER-COMPILER

A SOFTWARE TOOL FOR COMPILER CONSTRUCTION. COMPILER-COMPILERS ARE USED TO DEVELOP NEW COMPILERS WHEN THE HIGH LEVEL SOURCE LANGUAGE IS CHANGED OR A TOTALLY NEW SOURCE LANGUAGE IS ADOPTED.

COMPLEXITY

A MEASURE OF THE DIFFICULTY OF IMPLEMENTING A COMPONENT, INDEPENDENT OF THE IMPLEMENTOR'S EXPERIENCE. EASY (OR SIMPLE) MEANS THAT ANY GOOD PROGRAMMER CAN WRITE DOWN THE CORRECT CODE WITH LITTLE THOUGHT. HARD (OR COMPLEX) MEANS THAT MUCH THOUGHT IS INVOLVED IN THE DESIGN. (COMPARE THIS WITH "PRECISE"; E.G. EASY AND IMPRECISE MAY MEAN A VAGUE SPECIFICATION, BUT ONCE THE APPROACH IS DECIDED UPON, THE CODE IS EASY TO WRITE.) (SEL) (2) A TERM WHICH CAN REFER TO ANY NUMBER OF ASPECTS OF A COMPUTER PROGRAM: THE TOPOLOGY OF ITS CONTROL LOGIC, THE INTRICACY OF ITS DATA STRUCTURES, THE NUMBER OF COMPUTATIONS TO REACH AN ANSWER, THE SIZE OF THE PROGRAM, THE UNDERSTANDABILITY OF THE DOCUMENTATION, THE DEMONSTRATION EFFORT REQUIRED TO JUDGE CORRECTNESS, THE EASE WITH WHICH REPAIRS OR CHANGES CAN BE EFFECTED, ETC. (DAN 288) (3) CHARACTERISTICS OF A PROGRAM WHICH AFFECT COMPLEXITY INCLUDE: INSTRUCTION MIX, DATA REFERENCE, STRUCTURE/CONTROL FLOW, INTERACTION/INTERCONNECTION. (4) THE DEGREE OF INTERACTIONS AND DEPENDENCIES AMONG ELEMENTS OF A COMPUTER PROGRAM. (NASA)

COMPLEXITY MEASUREMENT

(1) THE PROCESS OF QUANTIFYING THE COMPLEXITY OF A PROGRAM. (2) THE NUMERICAL DESCRIPTION OF COMPLEXITY PRODUCED BY A MODEL OR FORMULA.

COMPLEXITY OF A PROGRAM

THE MINIMUM (CONCEPTUAL) LENGTH OF THE "PROOF OF CORRECTNESS" OF A PROGRAM, RELATIVE TO A PARTICULAR SET OF AVAILABLE METHODS FOR PERFORMING THE "PROOF OF CORRECTNESS", SUCH AS FORMAL MATHEMATICAL RIGOROUS THEOREM PROVING, INFORMAL (BUT COMPLETE) REASONING, EXHAUSTIVE TESTING, ETC. (DAN 1153)

COMPONENT

A COMPONENT IS A PIECE OF THE SYSTEM IDENTIFIED BY NAME OR COMMON FUNCTION (E.G., SEPARATELY COMPILABLE FUNCTION, AN ENTRY IN A TREE CHART OR BASELINE DIAGRAM FOR THE SYSTEM AT ANY POINT IN TIME, OR A SHARED SECTION OF DATA SUCH AS A COMMON BLOCK). (SEL)

COMPRESSION RATIO
THE MEASURE OF THE DEGREE OF COMPRESSION OF DATA AS EXPRESSED BY THE FRACTION: LENGTH OF ORIGINAL DATA/LENGTH OF COMPRESSED DATA. (DAN 781)

COMPUTATION ERROR/FAULT
AN ERROR/FAULT IN SOME ASSIGNMENT STATEMENT WHICH CAUSES THE WRONG FUNCTION TO BE COMPUTED FOR ONE OR MORE OF THE OUTPUT VARIABLES EVEN THOUGH THE SPECIFIC INPUT FOLLOWS THE CORRECT PATH. (DAN 842)

COMPUTATION STRUCTURE
AN ANALYTICAL TECHNIQUE USED TO MODEL THE DYNAMIC PERFORMANCE OF A COMPUTATION AND THE RESOURCES NEEDED TO PERFORM THE COMPUTATION. A COMPUTATION STRUCTURE CONSISTS OF TWO DIRECTED GRAPHS, A DATA FLOW GRAPH AND A PRECEDENCE GRAPH. THE DATA FLOW GRAPH ILLUSTRATES THE RELATIONSHIP BETWEEN THE STORAGE CELLS REQUIRED BY THE COMPUTATION AND THE SET OF OPERATORS WHICH MAY BE USED TO PROCESS THE INFORMATION CONTAINED IN THE CELLS. THE PRECEDENCE GRAPH INDICATES THE ORDER IN WHICH THESE OPERATIONS MUST BE EXECUTED IN ORDER TO CARRY OUT A COMPUTATION. THE TECHNIQUE IS USED TO EVALUATE THE PERFORMANCE OF DIFFERENT REALIZATIONS OF A COMPUTATION AND TO HELP SELECT THE ONE REALIZATION WHICH OFFERS THE BEST PERFORMANCE CHARACTERISTICS UNDER A GIVEN COST CONSIDERATION. (DAN 1127)

COMPUTER
A COMPUTER IS A MACHINE FOR CARRYING OUT CALCULATIONS OR TRANSFORMATIONS UNDER CONTROL OF A STORED PROGRAM...THE ABOVE DEFINITION IS A SELECTION FOR THE SOFTWARE ENGINEERING ENVIRONMENT FROM MORE LOOSELY FORMULATED ALTERNATIVE DEFINITIONS IN THE REFERENCE. IT APPLIES TO DIGITAL, ANALOG, AND HYBRID COMPUTERS. (SET)

COMPUTER ARCHITECTURE
THE SPECIFICATION OF THE RELATIONSHIPS BETWEEN THE PARTS OF A COMPUTER SYSTEM. (ANSI-X3) (2) THE MACHINE INSTRUCTION LEVEL OF A COMPUTER (DAN 286) (3) THE STRUCTURAL AND FUNCTIONAL DEFINITION OF A COMPUTER AS VIEWED IN TERMS OF ITS MACHINE INSTRUCTION SET AND INPUT/OUTPUT CAPABILITIES.(FISCA)

COMPUTER COMMUNICATION NETWORK
INDEXING TERM. REFERS TO THE SOFTWARE COMPONENT OF A COMPUTER COMMUNICATION NETWORK, OR TO THE USE OF SOFTWARE AS A TOOL IN THE DEVELOPMENT OF A COMPUTER COMMUNICATION NETWORK.

COMPUTER DATA
A REPRESENTATION OF FACTS, CONCEPTS OR INSTRUCTIONS IN A STRUCTURED COMMUNICATION BETWEEN COMPUTER EQUIPMENT. SUCH DATA CAN BE EXTERNAL (IN COMPUTER-READABLE FORM) OR RESIDENT WITHIN THE COMPUTER EQUIPMENT AND CAN BE IN THE FORM OF ANALOG OR DIGITAL SIGNALS. (DAN 158)

COMPUTER EQUIPMENT / COMPUTER HARDWARE
DEVICES CAPABLE OF ACCEPTING AND STORING COMPUTER DATA, EXECUTING A SYSTEMATIC SEQUENCE OF OPERATIONS ON COMPUTER DATA OR PRODUCING COMPUTER OUTPUTS. SUCH DEVICES CAN PERFORM SUBSTANTIAL INTERPRETATION, COMPUTATION, COMMUNICATION, CONTROL AND OTHER LOGICAL FUNCTIONS. EXAMPLES: CENTRAL PROCESSING UNITS, TERMINALS, PRINTERS, ANALOG/DIGITAL CONVERTERS, TAPE DRIVES, DISKS AND DRUMS. (DAN 158)

COMPUTER LOADING ANALYSIS
A SOFTWARE MANAGEMENT TOOL WHICH ANALYZES REQUIREMENTS VERSUS CAPABILITIES FOR BASIC PARAMETERS. TWO AVAILABLE TECHNIQUES ARE HAND ANALYSIS AND A COMPLEX COMPUTER PROGRAM, GPSS (GENERAL PURPOSE SIMULATION SYSTEM; E.G. IMB OR UNIVAC). (DAN 300)

COMPUTER NETWORK
(ISO) A COMPLEX CONSISTING OF TWO OR MORE INTERCONNECTED COMPUTERS. (ANSI-X3H1)

COMPUTER PROGRAM
A COMPUTER PROGRAM IS A SERIES OF INSTRUCTIONS OR STATEMENTS IN A FORM ACCEPTABLE TO COMPUTER EQUIPMENT DESIGNED TO CAUSE THE EQUIPMENT TO EXECUTE AN OPERATION OR OPERATIONS. (2) AN IDENTIFIABLE SERIES OF INSTRUCTIONS, OR STATEMENTS IN A FORM SUITABLE FOR EXECUTION BY A COMPUTER, PREPARED TO ACHIEVE A CERTAIN RESULT. (ANSI) DRAFT STANDARD FOR COMPUTER PROGRAM ABSTRACTS.

COMPUTER PROGRAM ABSTRACTS
A COMPUTER PROGRAM ABSTRACT IS A DESCRIPTIVE SUMMARY OF INFORMATION CONCERNING A COMPUTER PROGRAM. A COMPUTER PROGRAM ABSTRACT IS INTENDED TO PROVIDE SUFFICIENT INFORMATION FOR POTENTIAL USERS TO DETERMINE THE APPROPRIATENESS OF THE COMPUTER PROGRAM TO THEIR NEEDS AND RESOURCES.(ANSI-X3)

COMPUTER PROGRAM CERTIFICATION
COMPUTER PROGRAM CERTIFICATION IS THE PROCESS OF CONFIRMING THAT A COMPLETE COMPUTER PROGRAM IS OPERATIONALLY EFFECTIVE AND CAPABLE OF SATISFYING REQUIREMENTS UNDER SPECIFIED OPERATING CONDITIONS. COMPUTER PROGRAM CERTIFICATION USUALLY TAKES PLACE IN THE FIELD UNDER REAL CONDITIONS, AND IS UTILIZED TO EVALUATE NOT ONLY THE SOFTWARE ITSELF, BUT ALSO THE SPECIFICATIONS TO WHICH THE SOFTWARE WAS CONSTRUCTED. CERTIFICATION EXTENDS THE PROCESS OF VERIFICATION AND VALIDATION TO A REAL OR SIMULATED OPERATIONAL ENVIRONMENT. HERE THE CODE CAN BE EXERCISED TO DETERMINE WITH SOME CONFIDENCE WHETHER OR NOT THE STATED REQUIREMENTS ARE MET. OFFICIAL ENDORSEMENT OF THE OPERATIONAL CAPABILITY CAN THEN BE GIVEN. CERTIFICATION INVOLVES ACCEPTANCE TESTING OF THE OVERALL SYSTEM AND IS USUALLY ACCOMPLISHED BY OPERATIONAL TESTING, LABORATORY TESTING, AND/OR PLACING THE SYSTEM IN SIMULATED OPERATION. (SET) (2) THE TEST AND EVALUATION OF THE COMPLETE COMPUTER PROGRAM AIMED AT ENSURING OPERATIONAL EFFECTIVENESS AND SUITABILITY WITH RESPECT TO MISSION REQUIREMENTS UNDER REALISTIC OPERATING CONDITIONS. (DAN 134)

COMPUTER PROGRAM DEVELOPMENT PLAN (CPDP)
A MANAGEMENT PLAN. AIR FORCE REGULATION 800-14 DICTATES THAT COMPUTER PROGRAMS SHALL BE PLANNED, ANALYZED, DESIGNED, CODED, CHECKED, INTEGRATED, TESTED, AND DELIVERED IN ACCORDANCE WITH A CPDP. (DAN 356)

COMPUTER PROGRAM VALIDATION
THE TEST AND EVALUATION OF THE COMPLETE COMPUTER PROGRAM AIMED AT ENSURING COMPLIANCE WITH THE FUNCTION, PERFORMANCE, AND INTERFACE REQUIREMENTS. THE TESTS INCLUDE SYSTEM TESTS, SUBSYSTEM TESTS, AND INTEGRATION TESTS. THE AMOUNT OF TESTING DEPENDS UPON BOTH THE THOROUGHNESS OF THE SPECIFICATION DOCUMENT AND THE PARTICULAR VALIDATION PROCEDURES EMPLOYED. (SET)

COMPUTER PROGRAM VERIFICATION

THE TEST AND EVALUATION OF THE COMPLETE COMPUTER PROGRAM AIMED AT ENSURING OPERATIONAL EFFECTIVENESS AND SUITABILITY WITH RESPECT TO PROJECT REQUIREMENTS UNDER REALISTIC OPERATING CONDITIONS. (2) COMPUTER PROGRAM VERIFICATION IS THE ITERATIVE PROCESS OF DETERMINING WHETHER OR NOT THE PRODUCT OF EACH STEP OF THE COMPUTER PROGRAM ACQUISITION PROCESS FULFILLS ALL REQUIREMENTS LEVIED BY THE PREVIOUS STEP. THESE STEPS ARE SYSTEM SPECIFICATION VERIFICATION, REQUIREMENTS VERIFICATION, SPECIFICATION VERIFICATION, AND CODE VERIFICATION. VERIFICATION IS THE GENERIC PROCESS ENCOMPASSING THE FOUR ACTIVITIES DESCRIBED BELOW. ...SYSTEM SPECIFICATION VERIFICATION: SYSTEM SPECIFICATION VERIFICATION (SYSVER) IS THE PROCESS OF DETERMING WHETHER THE STATED MISSION REQUIREMENTS HAVE BEEN CLEARLY AND CORRECTLY TRANSLATED INTO AN ACHIEVABLE NEXT LOWER LEVEL OF SPECIFICATION. DETAILED REQUIREMENTS ANALYSES ARE CONDUCTED TO CRITICALLY EVALUATE PROPOSED CONCEPTUAL APPROACHES TO SYSTEM MECHANIZATION. PRELIMINARY SYSTEM AND SUBSYSTEM RELATIONSHIPS ARE REVIEWED TO IDENTIFY SATISFACTION OF APPROPRIATE PERFORMANCE, FUNCTIONAL, AND OPERATIONAL REQUIREMENTS. REQUIREMENTS ARE SEGMENTED IN SUFFICIENT DETAIL TO DETERMINE IF THE IDENTIFIED DESIGN APPROACHES CAN FULLY REALIZE THEM. THE PRIMARY OBJECTIVE OF SYSVER IS TO REDUCE THE RISK ASSOCIATED WITH SYSTEM ACQUISITION BY PROVIDING THE ANALYSIS AND REVIEW NECESSARY TO ENSURE VIABILITY. SYSVER IS USALLY ACCOMPLISHED PRIOR TO CONTRACT AWARD AND DURING THE ONSET OF CONTRACT PERFORMANCE. BECAUSE OF THE ROLE SOFTWARE PLAYS, CRITICAL ANALYSES AND SIMULATION OF KEY MODULES USUALLY ARE NECESSARY. ...REQUIREMENTS VERIFICATION: REQUIREMENTS VERIFICATION (REQVER) IS THE PROCESS OF DETERMINING WHETHER OR NOT THE COMPUTER PROGRAM REQUIREMENTS REFLECT THE COMPUTER-APPLICABLE PORTION OF THE SYSTEM SPECIFICATION. ITS PRIMARY PURPOSE IS TO IDENTIFY AMBIGUOUS, ILL-DEFINED, AND INADEQUATE COMPUTER REQUIREMENTS EARLY IN THE ACQUISITION CYCLE. REQVER SEEKS TO DETERMINE IF THE CONCEPTUAL DESIGN WILL WORK. ITS INTENT IS TO VERIFY THAT EACH REQUIREMENT STATED IN SYSTEM SPECIFICATION IS CLEARLY TRANSLATED INTO SUBSYSTEM REQUIREMENTS THAT CAN BE MECHANIZED. SIMULATIONS, DOCUMENT RESEARCH, AND ANALYSIS ARE THE TECHNIQUES PRESENTLY ASSOCIATED WITH REQVER. ...SPECIFICATION VERIFICATION: SPECIFICATION VERIFICATION (SPECVER) IS THE PROCESS OF DETERMINING WHETHER OR NOT THE DESIGN SPECIFICATION FOR THE INDIVIDUAL COMPUTER PROGRAM MODULES REPRESENTS A CLEAR CONSISTENT, AND ACCURATE TRANSLATION OF THE COMPUTER PROGRAM REQUIREMENTS. SPECVER IS CONCERNED WITH DETERMINING IF THE RECOMMENDED DESIGN ACTUALLY WILL DO THE JOB. IT DOESN'T SEEK TO REDESIGN, BUT RATHER TO IDENTIFY INADEQUACIES. TYPICALLY, THE ACTIVITIES ASSOCIATED WITH SPECVER ARE DOCUMENT ANALYSIS, INDEPENDENT SIMULATION, MODEL AND LOGIC ANALYSIS AND REDERIVATION OF KEY ALGORITHMS. ...CODE VERIFICATION: CODE VERIFICATION (CODEVER) IS THE PROCESS OF DETERMINING WHETHER OR NOT THE ACTUAL CODE IS COMPLIANT WITH THE TECHNICAL DESCRIPTION OF THE COMPUTER PROGRAM SPECIFICATION. THE ANALYSIS PERFORMED IS VERY DETAILED AND SEEKS TO IDENTIFY ERRORS OR DISCREPANCIES THAT STEM FROM INCONSISTENT USE OF INSTRUCTIONS, INCORRECT LOGIC FLOW, INCOMPATIBLE INTERFACES, FAILURES TO MEET TIMING AND SIZING BUDGETS, AND/OR INACCURACIES IN SCALING OR CALCULATIONS. (SET)

COMPUTER RESOURCES

THE TOTALITY OF AVAILABLE AND USEFUL COMPUTER EQUIPMENT, PROGRAMS, DOCUMENTATION, SERVICES, SUPPLIES AND PERSONNEL. (NASA) (2) THE TOTAL OF COMPUTER CAPABILITIES, MEMORY, AND MASS STORAGE. (DAN 1201) (3) IN AN ALLOCATION SENSE "RESOURCES" MORE SPECIFICALLY IS ORIENTED TO AVAILABLE MEMORY AND ALLOWABLE RUNNING TIME TO ACCOMPLISH A SPECIFIC JOB OR PROJECT.

COMPUTER SOFTWARE
A COMBINATION OF ASSOCIATED COMPUTER PROGRAMS AND DATA REQUIRED TO COMMAND THE COMPUTER EQUIPMENT TO PERFORM COMPUTATIONAL OR CONTROL FUNCTIONS. DAN 158) (2) THE TERMS SOFTWARE AND COMPUTER SOFTWARE ARE USED INTERCHANGEABLY. (SET) (3) IN A MORE DIRECT ALLOCATION SENSE "RESOURCES" MORE SPECIFICALLY ARE

COMPUTER SYSTEM
A COMPUTER SYSTEM IS AN INTERACTING COLLECTION OF COMPUTER EQUIPMENT, COMPUTER PROGRAMS, AND COMPUTER DATA. (SET)

COMPUTER TIME
FOR BATCH USAGE, THIS IS THE BILLABLE TIME FOR ALL RUNS. FOR INTERACTIVE USAGE, IT IS THE NUMBER OF HOURS SPENT AT A TERMINAL.(SEL) CONTRAST WITH EXECUTION TIME. (2) COMPUTER TIME IN SIMULATION, THE TIME REQUIRED TO PROCESS THE DATA THAT REPRESENTS A PROCESS OR THAT REPRESENTS A PART OF A PROCESS. (ANSI X-3)

COMPUTER TURNAROUND TIME
THE TIME DIFFERENTIAL BETWEEN THE SUBMITTAL OF A JOB TO THE COMPUTER CENTER AND THE RETURN OF THE JOB RESULTS TO THE PROGRAMMER. (DAN 137)

CONCISENESS
CODE POSSESSES THE CHARACTERISTIC CONCISENESS TO THE EXTENT THAT EXCESSIVE INFORMATION IS NOT PRESENT. THIS IMPLIES THAT PROGRAMS ARE NOT EXCESSIVELY FRAGMENTED INTO MODULES, OVERLAYS, FUNCTIONS, AND SUBROUTINES, NOR THAT THE SAME SEQUENCE OF CODE IS REPEATED IN NUMEROUS PLACES, RATHER THAN DEFINING A SUBROUTINE OR MACRO,ETC (DAN239)

CONCURRENT
CONCURRENT PERTAINS TO THE OCCURRENCE IN PARALLEL OF TWO OR MORE EVENTS OR ACTIVITIES WITHIN THE SAME SPECIFIED INTERVAL OF TIME...CONCURRENCY IN SOFTWARE DEVELOPMENT IS AN ISSUE, BECAUSE IT DOES OFFER THE POTENTIAL FOR DECREASING THE DURATION OF A DEVELOPMENT PROJECT. HOWEVER, IT ALSO PRESENTS THE PROBLEMS ASSOCIATED WITH MANY SIMULTANEOUS ACTIVITIES. THESE PROBLEMS INCLUDE MORE INTERFACE REQUIREMENTS, COORDINATION OF MULTIPLE TASKS, AND INTERTASK DEPENDENCIES. ONE EXAMPLE WOULD BE TO HAVE THE DESIGN AND CODING TASKS COMPLETED IN PARALLEL. (SET). (2) OPERATING OR OCCURRING AT THE SAME TIME. RUNNING IN PARALLEL. (ANSI-X3H1)

CONCURRENT PASCAL
AN EXTENSION OF PASCAL DESIGNED SPECIFICALLY FOR THE DESIGN AND IMPLEMENTATION OF MULTI-PROGRAMMING OPERATING SYSTEMS. (DAN 389)

CONCURRENT PROCESSES
PROCESSES MAY EXECUTE IN PARALLEL ON MULTIPLE PROCESSORS OR ASYNCHRONOUSLY ON A SINGLE PROCESSOR. CONCURRENT PROCESSES MAY INTERACT WITH EACH OTHER DURING EXECUTION. INDIVIDUAL PROCESSES WITHIN A COLLECTION OF CONCURRENT PROCESSES MAY SUSPEND THEIR EXECUTION PENDING RECEIPT OF INFORMATION FROM ANOTHER OF THE PROCESSES. (ABBOTT)

CONCURRENT PRODUCTION PRINCIPLE
A METHOD IN WHICH THE FORMAL PRODUCTION OF SOFTWARE PROCEEDS WITH CONCURRENT ACTIVITIES AMONG DESIGN, CODING, TESTING, AND DOCUMENTATION. (DAN 1153)

CONCURRENT PROGRAMMING
IN CONCURRENT PROGRAMMING, PROCESSES MAY INTERACT BY COMMUNICATION OF DATA AND SYNCHRONIZATION OF ACTIONS. (DAN 273)

CONDITION VARIABLES
MONITORS WHICH GOVERN QUEUES HAVE LOCAL CONDITION VARIABLES WITH ASSOCIATED WAIT AND SIGNAL OPERATIONS. A PROCESS MAY BE DELAYED WITHIN A MONITOR PROCEDURE UNTIL SOME CONDITION BECOMES TRUE.

CONDITIONAL CONTROL STRUCTURE
IN PROGRAMMING THIS STRUCTURE ALLOWS ALTERNATE BRANCHING OF PROGRAM FLOW DEPENDING UPON THE FULFILLMENT OF SPECIFIED CONDITIONS. IN MOST LANGUAGES IT GENERALLY FITS THE FORMAT IF...THEN...ELSE....

CONDITIONAL JUMP
A CONDITIONAL JUMP IS THE TRANSFER OF THE COMMAND SEQUENCE, PROVIDED SPECIFIED CRITERIA ARE MET.(SET)

CONFIDENCE LEVEL
PERCENT PROBABILITY THAT A GIVEN NUMBER IS CORRECT. 100% MEANS THAT THE NUMBER IS KNOWN TO BE CORRECT WITH ABSOLUTE CERTAINTY; 0% MEANS THAT THE NUMBER MUST BE INCORRECT. (AN OUTPUT OF SOME RELIABILITY AND ERROR MODELS) (SEL) (2) THE PROBABILITY THAT A GIVEN STATEMENT CONCERNING A SET OF RANDOM VARIABLES OR A SEGMENT OF A RANDOM PROCESS WILL BE UPHELD, IF TESTED. (DAN 1153)

CONFIGURATION
THE COLLECTION OF INTERCONNECTED OBJECTS WHICH MAKE UP A SYSTEM OR SUBSYSTEM. (ANSI-X3H1) (2) THE TOTAL SOFTWARE MODULES IN A SOFTWARE SYSTEM OR HARDWARE DEVICES IN A HARDWARE SYSTEM AND THEIR INTERRELATIONSHIPS. (DAN 1201)

CONFIGURATION CONTROL
A METHODOLOGY CONCERNED WITH PROCEDURES FOR CONTROLLING THE CONTENTS OF A SOFTWARE SYSTEM. A WAY OF MONITORING THE STATUS OF SYSTEM COMPONENTS, PRESERVING THE INTEGRITY OF RELEASED AND DEVELOPING VERSIONS OF A SOFTWARE SYSTEM, AND CONTROLLING THE EFFECTS OF CHANGES THROUGHOUT THE SYSTEM. (DAN LD7) (2) A PROCESS BY WHICH A CONFIGURATION ITEM IS BASELINED, AND THEREAFTER, ONLY CHANGEABLE BY APPROVAL BY A CONTROLLING AGENCY. (DAN 1201)

CONFIGURATION MANAGEMENT
CONFIGURATION MANAGEMENT INVOLVES THE SYSTEMATIC AND DISCIPLINED APPLICATION OF THE PRINCIPLES OF GOOD TECHNICAL AND ADMINISTRATIVE PRACTICES TO ENSURE THAT ALL REQUIREMENTS ARE IDENTIFIED, EVALUATED, TRANSFORMED INTO AND MAINTAINED AS HARDWARE CONFIGURATION ITEMS AND SOFTWARE CONFIGURATION ITEMS. IT IS THE FUNCTION OF CONFIGURATION MANAGEMENT TO PROVIDE THE FRAMEWORK FOR TECHNICAL CONTROL AND STATUS ACCOUNTING DURING CONFIGURATION ITEM ACQUISITION OR MODIFICATION TO BEST DIRECT MAINTENANCE EFFORT AND TO MINIMIZE IMPACT OF MAINTENANCE AND TESTING ON OPERATIONAL SERVICE. (DAN 223) (2) ALL ACTIVITIES RELATED TO CONTROLLING THE CONTENTS OF A SOFTWARE SYSTEM. IT MONITORS THE STATUS OF SYSTEM COMPONENTS, PRESERVES THE INTEGRITY OF RELEASED AND DEVELOPING VERSIONS OF A SYSTEM, AND CONTROLS THE EFFECTS OF CHANGES THROUGHOUT THE SYSTEM. IT IS A PROCESS DEALING AS MUCH WITH PROCEDURES AS WITH TOOLS. (DAN LD7) (3) A DISCIPLINE APPLYING TECHNICAL AND

ADMINISTRATIVE DIRECTION AND SURVEILLANCE TO IDENTIFY AND DOCUMENT A CONFIGURATION ITEM, TO CONTROL CHANGES TO IT, AND TO REPORT STATUS OF CHANGE PROCESSING AND IMPLEMENTATION. (DAN 1201)

CONFINEMENT
THE PROCESS OF ENSURING THAT WHILE ACCESSING A FILE THROUGH A FILE SYSTEM, NO INFORMATION FROM THE FILE WILL BE TRANSMITTED TO THE OUTSIDE WORLD (UNPRIVILEDGED USERS). (DAN 278)

CONNECTIONS, CONNECTIVITY
THE SET OF ASSUMPTIONS THE REST OF A PROGRAM MAKES ABOUT A MODULE (OR OTHER PROGRAM SEGMENT). MODULES HAVE CONNECTIONS IN CONTROL, IN DATA, AND IN SERVICES (FUNCTIONS) PERFORMED. CONNECTIVITY INCREASES WITH THE NUMBER, TYPE, AND VARIABILITY OF SUCH ASSUMPTIONS. (DAN 1153)

CONSISTENCY
THE STRICT AND UNIFORM ADHERENCE TO PRESCRIBED SYMBOLS, NOTATION, TERMINOLOGY, AND CONVENTIONS WHICH TENDS TO FOSTER A QUALITY SOFTWARE PRODUCT. (NASA) (2) A PROGRAM QUALITY WHICH ASSURES THAT THE RESULTS OF EXECUTING A PROGRAM ARE REPEATABLE IN A PRACTICAL SENSE, IN SPITE OF ANY LOGICAL ERRORS WHICH MAY BE PRESENT IN THE PROGRAM. (DAN 1153)

CONSISTENCY CHECKER
A COMPUTER PROGRAM USED TO DETERMINE (1) IF REQUIREMENTS AND/OR DESIGNS SPECIFIED FOR COMPUTER PROGRAMS ARE CONSISTENT WITH EACH OTHER AND (2) IF THEY ARE COMPLETE.

CONSISTENT
A COMPUTER PROGRAM IS INTERNALLY CONSISTENT TO THE EXTENT THAT IT CONTAINS UNIFORM NOTATION, TERMINOLOGY, AND SYMBOLOGY WITHIN ITSELF, AND IS EXTERNALLY CONSISTENT TO THE EXTENT THAT ITS FUNCTIONS ARE DIRECTLY RELATABLE TO THE REQUIREMENTS...SOME TESTS OF INTERNAL CONSISTENCY ARE: (A) CODING STANDARDS HOMOGENEOUSLY ADHERED TO: E.G., COMMENTS SHOULD NOT BE UNNECESSARILY EXTENSIVE OR WORDY AT ONE PLACE AND INSUFFICIENTLY INFORMATIVE IN ANOTHER. IRREGULAR USE OF UNEXPECTED OR NON-STANDARD CONSTRUCTIONS SHOULD BE AVOIDED; E.G., ABS(X) RATHER THAT AMAX1 (X,0.) - AMIN1 (X,0). (B) NAMES OF VARIABLES UNIQUE (IF RENAMED, THEN A CONSISTENT RELATIONSHIP SHOULD BE FOLLOWED). FOR EXAMPLE, USE PREFIX CHARACTER X- TO CONVERT INTEGER TO FLOATING-POINT REPRESENTATION (XNAME=NAME). (C) NUMBER OF ARGUMENTS IN SUBROUTINE CALLS MATCH WITH SUBROUTINE HEADER. (D) SINGLE, DOUBLE, OR MULTIPLE PRECISION REPRESENTATION USED CONSISTENTLY. TOLERANCES CONSISTENT WITH NUMBER OF SIGNIFICANT DIGITS IN INPUTS AND OUTPUTS. SOME TESTS OF EXTERNAL CONSISTENCY ARE: (A) EACH TEST DESCRIBED IN THE TEST PLAN IS DIRECTLY RELATABLE TO PROGRAM SPECIFICATION AND/OR REQUIREMENTS. (B) THERE IS A ONE TO ONE RELATIONSHIP BETWEEN FUNCTIONAL FLOW CHART ENTITIES AS DESCRIBED IN THE DETAILED DESIGN SPECIFICATION TO CODED ROUTINES OR MODULES OF A COMPUTER PROGRAM. (C) VARIABLE NAMES AND DEFINITIONS IN COMPUTER PROGRAM CODE, INCLUDING PHYSICAL UNITS, ARE CONSISTENT WITH GLOSSARY. (SET)

CONSTANTS AUTO CHECKER
A COMPUTER PROGRAM USED TO SEARCH A TAPE FOR ALL CONSTANTS AND PARAMETERS TO IDENTIFY THE NAME OF THE CONSTANT,ITS STORAGE LOCATION, AND THE BINARY SCALE FACTOR. THESE ARE THEN COMPARED WITH SPECIFICATION VALUES TO ASSURE COMPLIANCE. (DAN 134)

CONSTRAINT

CONSTRAINTS - RESTRICTIONS ON RESOURCE AVAILABILITY IMPOSED BY SPECIFICATIONS. SPACE CONSTRAINTS - ALL RESTRICTIONS OWING TO SPACE PROBLEMS, E.G., MAXIMUM NUMBER OF WORDS THAT COMPONENT MAY OCCUPY AT ONE TIME, MAXIMUM DISK SPACE AVAILABLE DURING EXECUTION TIME OR FOR PROGRAM STORAGE, ETC.. TIME CONSTRAINTS - ALL RESTRICTIONS OWING TO VARIOUS MACHINE AND CALENDAR TIME PROBLEMS, E.G., MAXIMUM EXECUTION TIME FOR COMPONENT TO PROCESS AND RESPOND TO SOME INPUT CONDITION, TIME TO COMPLETE A COMPONENT OR MILESTONE,ETC. (SEL)

CONTROL

A MAJOR SUB-DIVISION WITHIN CONFIGURATION MANAGEMENT. THE PROCEDURES BY WHICH CHANGES TO THE DESIGN REQUIREMENTS ARE PROPOSED AND FORMALLY PROCESSED. (DAN LD7).

CONTROL DATA

DATA THAT SELECTS AN OPERATING MODE OR SUBMODE IN A PROGRAM, DIRECTS THE SEQUENTIAL FLOW, OR OTHERWISE DIRECTLY INFLUENCES THE FUNCTION OF A PROGRAM. (DAN 1153)

CONTROL LOGIC

THE TOPOLOGICAL CONNECTIVITY AND THE SET OF CONDITIONS THAT TOGETHER GOVERN THE APPARENT SEQUENCING OF OPERATIONS WITHIN A PROCESS (OR AMONG CONCURRENT PROCESSES). CONTROL LOGIC IS OFTEN DISPLAYED BY MEANS OF A FLOWCHART. (DAN 1153)

CONTROL SEGMENT

A COLLECTION OF OPERATIONS AND OTHER CONTROL SEGMENTS ORGANIZED ACCORDING TO A SINGLE CONTROL STRUCTURE. THE PARTICULAR OPERATIONS SELECTED FROM A CONTROL SEGMENT AND THE ORDER OF THEIR PERFORMANCE DURING THE EXECUTION OF A CONTROL SEGMENT MAY BE DETERMINED COMPLETELY FROM THE INFORMATION AVAILABLE IN THE CONTROL SEGMENT AND THE DATA OBJECTS TO WHICH THE CONTROL SEGMENT IS APPLIED. THIS PROPERTY OF CONTROL SEGMENTS IS THE PROPERTY OF HAVING A SINGLE ENTRY. ON COMPLETION OF EXECUTION OF A CONTROL SEGMENT, NO EXECUTION REQUIREMENTS ARE LEFT PENDING OR INCOMPLETE. THERE ARE NO SPECIFICATIONS WITHIN A CONTROL SEGMENT INDICATING WHICH OTHER CONTROL SEGMENTS ARE TO BE EXECUTED AFTERWARDS. THIS PROPERTY OF CONTROL SEGMENTS IS THE PROPERTY OF HAVING A SINGLE EXIT. (ABBOTT)

CONTROL STATEMENTS

ALL STATEMENTS THAT POTENTIALLY ALTER THE SEQUENCE OF EXECUTED INSTRUCTIONS (E.G., GOTO, IF, RETURN, DO). (SEL) (2) COMPARE WITH CONTROL STRUCTURES. (3) A STATEMENT IN A PROGRAMMING LANGUAGE WHICH, WHEN EXECUTED, AFFECTS THE ORDER IN WHICH (OTHER) OPERATIONS ARE EXECUTED. NOT ALL CONTROL STATEMENTS DEFINE CONTROL STRUCTURES. (ABBOTT)

CONTROL STRUCTURES

CONTROL STRUCTURES ARE THE LOGICAL EXPRESSIONS THAT DETERMINE THE FLOW OF CONTROL THROUGH A COMPUTER PROGRAM... STRUCTURED PROGRAMMING RESTRICTS FLOW OF CONTROL CONSTRUCTS TO SIMPLE STRUCTURES AND AVOIDS TRANSFERS OF CONTROL THAT CREATE FLOW COMPLEXITIES (I.E., EXCESSIVE GOTO STATEMENTS). SET) (2) AN ORGANIZATION USED TO BUILD A CONTROL SEGMENT. A CONTROL STRUCTURE RELATES TWO OR MORE OPERATIONS OR CONTROL SEGMENTS WITHIN AN ALGORITHM. A CONTROL STRUCTURE PROVIDES THE FRAMEWORK TO DETERMINE: 1) WHETHER ITS COMPONENT

OPERATIONS AND CONTROL SEGMENTS WILL BE PERFORMED; AND 2) THE ORDER IN WHICH THEY WILL BE PERFORMED DURING EXECUTION OF AN ALGORITHM. (ABBOTT)

CONVENTION
AN AGREED METHOD, FORM OF PRESENTATION TO PROVIDE CONSISTENCY AND UNDERSTANDING TO DELIVERABLE SOFTWARE ELEMENTS. (DAN 1201)

CONVERSION AIDS
THOSE SOFTWARE TOOLS WHICH ASSIST IN CONVERTING OPERATIONAL SOFTWARE FROM ONE COMPILER TO ANOTHER. THESE TOOLS ANALYZE THE SOURCE CODE AS WRITTEN FOR ONE COMPILER AND HIGHLIGHT THOSE STATEMENTS WHICH ARE NOT COMPATIBLE WITH THE CAPABILITIES OF THE TARGET COMPILER. IN SOME INSTANCES THESE CONVERSION AIDS WILL REPLACE THE INCOMPATIBLE STATEMENTS WITH ONE OR MORE TARGET COMPILER STATEMENTS WHICH ARE DESIGNED TO ACHIEVE THE SAME RESULT. (DAN 142)

CONVERSION COST FACTORS
SPECIFIC INFORMATION CONCERNING ONE OR MORE COST FACTORS INCURRED DURING A SOFTWARE CONVERSION. (DAN 786)

CONVERSION COSTS
COSTS RELATED TO OR INCURRED DURING A SOFTWARE CONVERSION EFFORT.(DAN 786)

CONVERSIONS
THIS TERM REFERS TO THE CONVERSION OF EXISTING SOFTWARE FROM ONE LANGUAGE TO ANOTHER LANGUAGE OR FROM ONE HARDWARE/SOFTWARE CONFIGURATION TO ANOTHER. (DAN 786)

COPY
(ISO) TO READ DATA FROM A SOURCE, LEAVING THE SOURCE DATA UNCHANGED, AND TO WRITE THE SAME DATA ELSEWHERE IN A PHYSICAL FORM THAT MAY DIFFER FROM THAT OF THE SOURCE. A COPY DIFFERS FROM A MOVE IN THAT IT PRESERVES THE SOURCE UNCHANGED. (ANSI-X3H1)

COROUTINES
COROUTINES ARE TWO COMPUTER PROGRAMS WHICH CAN CALL ON EACH OTHER...SUBROUTINES ARE SPECIAL CASES OF MORE GENERAL PROGRAM COMPONENTS CALLED COROUTINES. IN CONTRAST TO THE SUBORDINATE RELATIONSHIP OF A SUBROUTINE TO A MAIN ROUTINE, THERE IS COMPLETE SYMMETRY BETWEEN COROUTINES WHICH CALL ON EACH OTHER. IT IS NOT NECESSARY THAT EACH CALL RESULTS IN THE COMPLETE EXECUTION OF THE OTHER.(SET)

CORRECTION
A CHANGE MADE TO CORRECT AN ERROR. (SEL)

CORRECTIVE MAINTENANCE
MAINTENANCE SPECIFICALLY INTENDED TO ELIMINATE AN EXISTING FAULT... CONTRAST WITH PREVENTIVE MAINTENANCE. (ANSI-X3)

CORRECTIVE MAINTENANCE TIME
TIME, EITHER SCHEDULED OR UNSCHEDULED, USED TO PERFORM CORRECTIVE MAINTENANCE. (ANSI-X3)

CORRECTNESS
AGREEMENT BETWEEN A PROGRAM'S TOTAL RESPONSE AND THE STATED RESPONSE IN THE

FUNCTIONAL SPECIFICATION (FUNCTIONAL CORRECTNESS), AND/OR BETWEEN THE PROGRAM AS CODED AND THE PROGRAMMING SPECIFICATION (ALGORITHMIC CORRECTNESS). (DAN 1153)

CORRECTNESS PROOFS
PROOF THAT A PROGRAM PRODUCES CORRECT RESULTS FOR ALL POSSIBLE INPUTS. VALIDATION OF A PROGRAM IN THE SAME WAY A MATHEMATICAL THEOREM IS PROVED CORRECT. I.E., BY MATHEMATICAL ANALYSIS OF ITS PROPERTIES. (DAN LD7) (2) AN ALTERNATIVE TO EXECUTING TESTS OF SOFTWARE TO DEMONSTRATE ITS CORRECTNESS IS THE METHOD OF ANALYTIC PROOFS. THE VERIFICATION PROCESS CONSISTS OF MAKING ASSERTIONS DESCRIBING THE STATE OF A PROGRAM INITIALLY, AT INTERMEDIATE POINTS IN THE PROGRAM FLOW, AND AT TERMINATION, AND THEN PROVING THAT EACH ASSERTION IS IMPLIED BY THE INITIAL OR PRIOR ASSERTION AND ALSO BY THE TRANSFORMATIONS PERFORMED BY THE PROGRAM BETWEEN EACH TWO CONSECUTIVE ASSERTIONS. AN ASSERTION CONSISTS OF A DEFINITION OF THE RELATIONSHIPS AMONG THE VARIABLES AT THE POINT IN THE PROGRAM WHERE THE ASSERTION IS MADE. THE PROOFS EMPLOY STANDARD TECHNIQUES FOR PROVING THEOREMS IN THE FIRST ORDER PREDICATE CALCULUS. PROOF OF THE CORRECTNESS OF A PROGRAM USING THIS APPROACH OBVIATES THE NEED FOR EXECUTING TEST CASES, SINCE ALL POSSIBILITIES ARE COVERED BY THE PROOFS. (DAN 172) (3) THE TECHNIQUE OF PROVING MATHEMATICALLY THAT A GIVEN PROGRAM IS CONSISTENT WITH A GIVEN SET OF SPECIFICATIONS. THIS PROCESS CAN BE ACCOMPLISHED BY MANUAL METHODS OR BY PROGRAM VERIFIERS REQUIRING MANUAL INTERVENTION. (DAN 154) (4) AUTOMATED VERIFICATION SYSTEMS EXIST WHICH ALLOW THE ANALYST TO PROVE SMALL PROGRAMS ARE CORRECT BY MEANS SIMILAR TO THOSE USED IN PROVING MATHEMATICAL THEOREMS. AXIOMS AND THEOREMS DERIVED ARE USED TO ESTABLISH VALIDITY OF PROGRAM ASSERTIONS AND TO PROVIDE A FUNDAMENTAL UNDERSTANDING OF HOW THE PROGRAM OPERATES. (DAN 134)

COSMETIC
CHANGES IN THE SOURCE PROGRAM THAT HAVE LITTLE EFFECT ON THE PERFORMANCE OF PROGRAM. (E.G., CORRECT COMMENTS, MOVE CODE AROUND AS LONG AS IT DOES NOT ALTER THE ALGORITHM IMPLEMENTED, CHANGE THE NAME OF A LOCAL VARIABLE, ETC. (SEL)

COST AND SCHEDULE CONTROL
INDEXING TERM. REFERS TO MANAGEMENT TOOLS AND/OR TECHNIQUES WHICH CAN BE USED TO EFFECT COST AND SCHEDULE CONTROL.

COST DATA
DATA DESCRIBING THE COSTS ASSOCIATED WITH THE RESOURCES EXPENDED BY A SOFTWARE PROJECT. (DAN 137)

COST EFFECTIVE
A TERM WHICH DESCRIBES SOME METHOD, TOOL OR TECHNIQUE THAT REDUCES THE PREDICTED COST TO PERFORM ON A CONTRACTED ITEM. (DAN 1201)

COST ESTIMATION
A STANDARD TECHNIQUE FOR ESTIMATING THE AMOUNT OF LABOR NECESSARY FOR THE COMPLETION OF A TASK, THE AMOUNT AND POTENTIAL COSTS OF COMPUTER TIME REQUIRED, ETC., PRIOR TO AND DURING A PROJECT'S LIFETIME. (DAN LD7)

COST FACTORS
IDENTIFIED PARAMETERS, CONSTRAINTS OR SYSTEM CHARACTERISTICS WHICH AFFECT

THE MAGNITUDE OR DISTRIBUTION OF COSTS DURING THE SOFTWARE LIFE CYCLE. (DAN 772)

COST MANAGEMENT
A COLLECTED SET OF TOOLS THAT PROVIDE THE CRITERIA AND DEVICES FOR TRACING PROJECT COSTS. (DAN LD7)

COSTING TECHNIQUES
METHODS FOR DETERMINING THE COST OF DEVELOPING A SYSTEM OR ANY PARTICULAR PART OF A SYSTEM.

COST-BENEFIT ANALYSIS
COST-BENEFIT ANALYSIS SEEKS TO ESTIMATE AND COMPARE THE COSTS AND BENEFITS OF AN UNDERTAKING. IT CAN BE USED IN ANY OR ALL OF THREE WAYS: (1) AS A PLANNING TOOL FOR ASSISTANCE IN CHOOSING AMONG ALTERNATIVES AND ALLOCATING (SCARCE) RESOURCES AMONG COMPETING DEMANDS, (2) AS AN AUDITING TOOL FOR PERFORMING POST HOC EVALUATIONS OR FOLLOW-UP STUDIES OF AN EXISTING PROJECT; (3) AS A WAY TO DEVELOP "QUANTITATIVE" SUPPORT IN ORDER TO POLITICALLY INFLUENCE A DECISION. (4) AS A METRIC TO ESTIMATE EFFECTIVENESS OF A PROPOSED SOFTWARE TOOL OR TO COMPARE PROPOSED OR EXISTING SOFTWARE TOOLS FOR A GIVEN PROJECT. (DAN 415)

COSTS
SEE COSTING TECHNIQUES, COST AND SCHEDULE CONTROL, COST ESTIMATING, ERROR CORRECTION COSTS.

CREATE
THE CREATION OF THE IDEA AND THE RECORDING OF IT. (SEL)

CREATION DATE
DATE COMPONENT WAS FIRST NAMED (E.G., DATE IT FIRST APPEARED ON A TREE CHART). (SEL)

CRISP
(CONTROL-RESTRICTIVE INSTRUCTIONS FOR STRUCTURED PROGRAMMING) A SET OF KEYWORDS USED TO INTRODUCE STRUCTURED CONTROL FLOW INTO AN UNSTRUCTURED LANGUAGE. ALSO USED AS CONTROL SUBLANGUAGE OF CRISP-FLOW (FLOWCHARTS) AND CRISP-PDL PROCESSORS. (DAN 1153)

CRITICAL PIECE FIRST
THE IMPLEMENTATION OF THE MOST CRITICAL ASPECTS OF THE SYSTEM FIRST.

CRITICAL REGION
A REGION WITHIN A PROCESS IN WHICH A SHARED RESOURCE MUST ONLY BE ACCESSED ON A MUTUALLY EXCLUSIVE BASIS FOR PROGRAM CONSISTENCY AND CORRECTNESS. (DAN 1153)

CROSS COMPILER
A TOOL THAT CAN OPERATE ON A HOST COMPUTER AND PRODUCE CODE FOR A DESIGNATED EXTERNAL COMPUTER. (DAN LD7) (2) A COMPILER PROGRAM WHICH IS EXECUTED BY ONE COMPUTER TO GENERATE OBJECT CODE FOR ANOTHER TYPE OF COMPUTER. (NASA)

CROSS REFERENCE
A LIST OF THE IDENTIFIERS USED BY A PROGRAM SHOWING (BY MEANS OF INDICES OR

STATEMENT NUMBERS) WHICH STATEMENTS OF THE PROGRAM DEFINE AND REFERENCE THOSE IDENTIFIERS. (SEL) (2) A NOTATION OR DIRECTION IN ONE PLACE TO PERTINENT INFORMATION IN ANOTHER PLACE. OFTEN USED FOR AN INDEX OF VARIABLE NAMES IN A PROGRAM. (ANSI-X3H1)

CROSS-ASSEMBLER
A COMPUTER PROGRAM THAT ACCEPTS SYMBOLIC INSTRUCTION MNEMONICS FOR A SELECTED TARGET COMPUTER AND GENERATES TARGET COMPUTER MACHINE CODE WHILE HOSTED ON ANOTHER COMPUTER. A CROSS-ASSEMBLER THUS ALLOWS CODE WRITTEN FOR ONE COMPUTER TO BE ASSEMBLED ON ANOTHER. (DAN 134)

CROSS-REFERENCE PROGRAMS
A GROUP OF COMPUTER PROGRAMS THAT PROVIDE CROSS-REFERENCE INFORMATION ON SYSTEM COMPONENTS. FOR EXAMPLE, PROGRAMS CAN BE CROSS-REFERENCED WITH OTHER PROGRAMS, MACROS, PARAMETER NAMES, ETC. THIS CAPABILITY IS USEFUL IN PROBLEM-SOLVING AND TESTING TO ASSESS IMPACT OF CHANGES TO ONE AREA OR ANOTHER. (DAN 134) (2) UTILITY PROGRAMS WHICH PROVIDE CROSS-REFERENCE DATA CONCERNING A PROGRAM WRITTEN IN A HIGHER LEVEL LANGUAGE. THESE UTILITY PROGRAMS ANALYZE A SOURCE PROGRAM AND PROVIDE AS OUTPUT SUCH DATA AS FOLLOWS: 1.STATEMENT LABEL CROSS-INDEX 2. DATA NAME CROSS-INDEX 3. LITERAL USAGE CROSS-INDEX 4. INTER-SUBROUTINE CALL CROSS-INDEX 5. STATISTICAL COUNTS OF STATEMENT TYPES (DAN 142)

CURRICULA
INDEXING TERM. REFERS TO COURSES OF STUDY IN SOFTWARE ENGINEERING.

CYCLIC DATA
DATA ON PROGRAMMING ACTIVITES THAT OCCURRED SINCE THE LAST MANAGEMENT REPORTING PERIOD. (DAN 137)

DAS (DESIGN ANALYSIS SYSTEM)
AN AUTOMATED SYSTEM THAT SUPPORTS DESIGN VERIFICATION WITH THE GOAL OF IMPROVED SOFTWARE QUALITY AND REDUCED LIFE CYCLE COSTS. (DAN 256)

DATA
DATA IS A REPRESENTATION OF FACTS, CONCEPTS, OR INSTRUCTIONS IN A STRUCTURED FORM SUITABLE FOR ACCEPTANCE, INTERPRETATION, OR PROCESSING BY COMPUTER EQUIPMENT...DATA CAN BE EXTERNAL (IN COMPUTER-READABLE FORM) OR RESIDENT WITHIN THE COMPUTER EQUIPMENT AND CAN BE IN THE FORM OF ANALOG OR DIGITAL SIGNALS. (SET) (2) A SET OF FACTS. (DAN 137)

DATA ANALYSIS
INDEXING TERM. REFERS TO THE APPLICATION OF STATISTICAL PROCEDURES TO RAW DATA TO OBTAIN INFORMATION RELATING TO SOME ASPECT OF SOFTWARE DEVELOPMENT,USE, RELIABILITY, OR MAINTENANCE.

DATA ANALYSIS TOOLS
PROGRAMS SPECIFICALLY DESIGNED TO PERFORM STATISTICAL AND COMPARATIVE ANALYSIS ON DATA PRODUCED DURING THE EXECUTION OF A PROGRAM TEST. (DAN LD7)

DATA BASE
(1) (ISO) A SET OF DATA, PART OR THE WHOLE OF ANOTHER SET OF DATA, AND CONSISTING OF AT LEAST ONE FILE, THAT IS SUFFICIENT FOR A GIVEN PURPOSE OR FOR A GIVEN DATA PROCESSING SYSTEM. (2) A COLLECTION OF DATA FUNDAMENTAL TO

A SYSTEM. (3) A COLLECTION OF DATA FUNDAMENTAL TO AN ENTERPISE. (ANSI-X3)

DATA BASE ANALYZER
A COMPUTER PROGRAM THAT REPORTS INFORMATION ON EVERY USAGE OF DATA, IDENTIFIES EACH PROGRAM USING ANY DATA ELEMENTS, AND INDICATES WHETHER THE PROGRAM INPUTS, USES, MODIFIES, OR OUTPUTS THE DATA ELEMENT. ANY UNUSED DATA IS PRINTED. ERRORS DEALING WITH MISUSE AND NON-USE OF DATA AND CONFLICTS IN DATA USAGE ARE IDENTIFIED. (DAN 134)

DATA BASE APPLICATIONS
THIS CATEGORY IS TO INCLUDE COMPONENTS WHICH RETRIEVE, WRITE TO, OR FORMAT INFORMATION FOR A WELL DEFINED FORMATTED BANK OF INFORMATION AVAILABLE TO THE SYSTEM. IT IS UP TO THE USER TO DECIDE WHETHER THE DATA SET IS TO BE CONSIDERED A DATA BASE OR NOT.AN EXAMPLE OF AN ACCEPTABLE DATA BASE WOULD BE THE ADL FILE, SLP FILE, GEODETICS FILE, ETC., WHILE A SEQUENTIAL TELEMETRY FILE ON TAPE WOULD NOT BE. (SEL)

DATA BASE MANAGEMENT SYSTEM
A DATA BASE MANAGEMENT SYSTEM (DBMS) INCLUDES A DATA DESCRIPTION LANGUAGE (DDL) FOR DESCRIBING THE LOGICAL ORGANIZATION OF DATA IN THE DATA BASE, AND A DATA MANIPULATION LANGUAGE (DML) FOR ACCESSING AND MODIFYING THE DATA BASE. (DAN 273)

DATA COLLECTION
REFERS TO THE METHODS (I.E. FORMS, PROCEDURES, PERSONNEL) FOR COLLECTING DATA AND THE POINT (TIME) AT WHICH DATA COLLECTION SHOULD BEGIN. (DAN 295)

DATA COLLECTION COSTS
THE COST OF COLLECTING DATA, ESPECIALLY WITH REGARD TO ERROR DATA. (DAN 509)

DATA DEFINITION LANGUAGE
A COMPUTER PROGRAM USED TO DESCRIBE DATA AT A SUFFICIENTLY HIGH LEVEL IN ORDER TO MAKE THE USE OF A PARTICULAR PROGRAMMING LANGUAGE TRANSPARENT TO THE DATA DEFINITION PROCESS. THIS LANGUAGE ALLOWS US TO SPECIFY THE DATA SO THAT MULIPLE LANGUAGES CAN SHARE AND USE IT. (DAN 134)

DATA DICTIONARY
A LISTING OF THE NAMES, LENGTHS AND REPRESENTATIONS OF ALL DATA ITEMS USED IN A SOFTWARE SYSTEM. THIS TOOL MAY BE MANUAL OR AUTOMATED.

DATA DOMAIN
AN APPROACH TO OBTAINING AN ESTIMATE OF OPERATIONAL RELIABILITY. IN PRINCIPAL, IF ALL SETS OF INPUT DATA VALUES UPON WHICH THE COMPUTER PROGRAM MUST OPERATE ARE IDENTIFIED, AN ESTIMATE OF THE RELIABILITY OF THE PROGRAM COULD BE OBTAINED BY RUNNING THE PROGRAM FOR ALL SUCH POSSIBLE SETS. IN PRACTICE A METHODOLOGY IS USED TO SELECT SAMPLE DATA SETS FROM THE TOTAL OF SUCH SETS WHICH ARE REPRESENTATIVE OF INTENDED OPERATIONAL USAGE AND THE PROGRAM IS RUN FOR THOSE SETS ONLY. THE RESULTS OF THE RUNS ON THE SAMPLE DATA SETS ARE USED TO COMPUTE RELIABILITY ESTIMATES FOR THE PROGRAM IN ITS INTENDED OPERATIONAL ENVIRONMENT. (DAN 238)

DATA FLOW DIAGRAMS
SEE DATA FLOWGRAPH

DATA FLOWGRAPH
A DEVICE WHICH HELPS TO GRAPHICALLY DISPLAY WHAT HAPPENS TO DATA, HOW DATA IS TRANSFORMED, AND HOW ONE CAN PARTITION THE PROCESS INTO SUBPROCESSES WITH A MINIMAL NEED OF DATA TRANSFERS. (DAN 323)

DATA ITEM
A SPECIFIC ENTITY OF DATA. (DAN 137)

DATA OBJECT
AMBIGUOUSLY, EITHER A NAME OR A VALUE TO WHICH AN OPERATION MAY BE APPLIED. (ABBOTT)

DATA PARAMETERS - RELIABILITY
THE INPUT(S) TO A RELIABILITY ESTIMATION PROGRAM. THESE INPUTS OR PARAMETERS ARE USUALLY GROUPED INTO FIVE CATEGORIES; (1) FAILURE DATA WHICH INCLUDES A SET OF EXECUTION TIME INTERVALS BETWEEN FAILURES, ALONG WITH THE NUMBER OF DAYS FROM THE START OF TESTING ON WHICH THE FAILURES OCCURRED; (2) PLANNED DATA INCLUDES AVAILABLE COMPUTER TIME, NUMBERS OF AVAILABLE FAILURE CORRECTION PERSONNEL AND FAILURE IDENTIFICATION PERSONNEL, COMPUTER TIME UTILIZATION FACTOR, FAILURE CORRECTION PERSONNEL UTILIZATION FACTOR, AND OBJECTIVE MEAN TIME TO FAILURE; (3) DEBUG ENVIRONMENT DATA INCLUDES AVERAGE OF COMPUTER TIME REQUIRED PER FAILURE, CORRECTION WORK REQUIRED PER FAILURE AND FAILURE IDENTIFICATION WORK; (4) TEST ENVIRONMENT DATA INCLUDES THE TESTING COMPRESSION FACTOR AND THE TIMES WHEN TESTING WAS STARTED, STOPPED, OR INTERRUPTED; AND (5) PROGRAM DATA INCLUDES THE ESTIMATION OF THE NUMBER OF FAILURES REQUIRED TO EXPOSE AND REMOVE ALL ERRORS, AND INITIAL MTTF AT START OF TESTING.

DATA PROCESSING APPLICATIONS
(ISO) THE EXECUTION OF A SYSTEMATIC SEQUENCE OF OPERATIONS PERFORMED UPON DATA, E.G. HANDLING, MERGING, SORTING, COMPUTING. SYNONYMOUS WITH INFORMATION PROCESSING. (ANSI-X3)

DATA REDUCTION TOOLS
PROGRAMS, OFTEN DATA-BASE DEPENDENT, WHICH PROCESS A DATA SET AND CONVERT THE INFORMATION INTO READABLE FORM. THESE PROGRAMS PERFORM STATISTICAL AND COMPARATIVE TRANSFORMATIONS ON THE RECORDED DATA OBTAINED BY INSTRUMENTATION. (DAN LD7)

DATA REPOSITORY
A FACILITY FOR GATHERING , STORING AND DISSEMINATING DATA RELATED TO A PARTICULAR TOPIC OR GROUP OF TOPICS.

DATA RESTRUCTURING
THE PROCESS OF CHANGING THE REPRESENTATION OF DATA IN MEMORY.

DATA SEMANTICS
FROM THE USERS' POINT OF VIEW, A DATABASE IS A COLLECTION OF INFORMATION MODELING SOME ENTERPRISE IN THE REAL WORLD. THE ROLE OF DATA SEMANTICS IS TO ENSURE THAT STORED DATA ACCURATELY REPRESENTS THE ENTERPRISE. (DAN 412)

DATA STRUCTURE
THE LOGICAL RELATIONSHIPS WHICH EXIST AMONG THE UNITS OF DATA IN A DATABASE AND WHICH ARE UNDER CONTROL OF A DATABASE MANAGEMENT SYSTEM. (2) A

FORMALIZED REPRESENTATION OF THE ORDERING AND ACCESSIBILITY RELATIONSHIPS AMONG STORED DATA ITEMS, WITHOUT REGARD TO THE ACTUAL STORAGE CONFIGURATION, AS CHARACTERIZED BY DATA-ITEM TYPES, RANGES OF VALUES, AND SCOPE OF ACTIVITY, SUITABLE FOR COMMUNICATION, INTERPRETATION, OR PROCESSING BY HUMAN OR AUTOMATIC MEANS. (DAN 1153)

DATA TYPE(S)
A DOMAIN OF VALUES AND AN INTERPRETATION FOR THOSE VALUES. (ANSI-X3H1) (2) ONE CLASS IN A DATA TYPOLOGY. ALL OBJECTS IN A DATA TYPE MAY BE SUBJECTED TO THE SAME OPERATIONS. (ABBOTT) (3) A SET OF ATTRIBUTES USED TO DEFINE A SET WHOSE ELEMENTS ARE DATA STRUCTURES, AND ON WHICH AN ALGEBRA IS DEFINED. FUNDAMENTAL TYPES ARE THOSE EXPLICIT IN A PROGRAMMING LANGUAGE. FUNDAMENTAL SIMPLE TYPES USUALLY INCLUDE INTEGERS AND REALS, AND FUNDAMENTAL STRUCTURES USUALLY INCLUDE THE INDEXED ARRAY. (DAN 1153)

DATA TYPOLOGY
A CLASSIFICATION FOR DATA OBJECTS TO BE MANIPULATED BY PROGRAMS. THE CLASSIFICATION SERVES TO ORGANIZE DATA OBJECTS INTO CLASSES (CALLED DATA TYPES) OVER WHICH OPERATIONS MAY BE DEFINED. (ABBOTT)

DATA VALIDATION
THE PROCESS OF VERIFYING THAT DATA WHICH HAS BEEN COLLECTED AND ENTERED INTO A DATA BASE IS ACCURATE AND COMPLETE.

DATA WORD
A BINARY NUMBER WITH A PRESCRIBED FORMAT AND NUMBER OF BITS. (NASA)

DATAWARE
ONE OF TWO MAJOR COMPONENTS OF SOFTWARE AS DEFINED BY T. GILB. DATAWARE IS THE PHYSICAL FORM IN WHICH ALL INFORMATION, INCLUDING LOGICWARE,APPEARS TO THE HARDWARE, AND WHICH IS PROCESSED AS A RESULT OF THE LOGIC OF THE LOGICWARE. SEE ALSO LOGICWARE. (DAN 781)

DEADLOCK
A STATE OF INACTION OR NEUTRALIZATION RESULTING FROM THE OPPOSITION OF TWO OR MORE EQUALLY POWERFUL ENTITIES. OFTEN USED WHEN RESOURCES ARE ALLOCATED TO VARIOUS ENTITIES, SUCH THAT NONE CAN PROCEED WITHOUT THE RESOURCES OF ANOTHER. DEADLOCK AVOIDANCE IS THE TECHNIQUE OF ALLOCATING RESOURCES SUCH THAT DEADLOCK CANNOT OCCUR. DEADLOCK DETECTION IS THE TECHNIQUE OF DETECTING DEADLOCK ONCE IT HAS OCCURRED. DEADLY EMBRACE IS SYNONYMOUS WITH DEADLOCK. (ANSI-X3H1)

DEADLY EMBRACE
SYNONYM FOR DEADLOCK

DEALLOCATE
THE CONVERSE OF ALLOCATE. (ANSI-X3H1)

DEBUGGING
TESTING IS THE PROCESS OF DETERMINING WHETHER OR NOT ERRORS/FAULTS EXIST IN A PROGRAM. DEBUGGING IS AN ATTEMPT TO ISOLATE THE SOURCE OF THE PROBLEM AND TO FIND A SOLUTION...DEBUGGING IS REQUIRED ONLY IN THE EVENT THAT ONE OR MORE TESTS FAIL. IT IS THE PROCESS OF LOCATING THE ERROR/FAULT WHICH CAUSED A TEST TO FAIL. (DAN 609) (2) THE IDENTIFICATION AND CORRECTION OF SOFTWARE

DISCREPANCIES.(NASA)

DEBUGGING MODEL
AMODLL WHICH RELATES THE EFFECTS OF BUG REMOVAL, SPAWNED BUGS, AND BUG DETECTION WITHOUT BUG REMOVAL DURING THE DEBUGGING/TESTING PHASE OF THE SOFTWARE LIFE CYCLE. A DEBUGGING MODEL'S OUTPUTS MAY INCLUDE ESTIMATES OF THE NUMBER OF REMAINING ERRORS/FAULTS AT A GIVEN TIME, TIME NECESSARY TO REACH A GIVEN LEVEL OF "ERROR FREENESS", OR COST OF DEBUGGING TO A GIVEN LEVEL OF "ERROR FREENESS".

DEBUGGING TOOLS
THOSE PROGRAMS DESIGNED TO LOCATE AND ELIMINATE PROGRAMMING ERRORS AND TO TEST A PROGRAM FOR PROPER EXECUTION . (DAN LD7) (2) SOFTWARE TOOLS AVAILABLE TO THE SYSTEM OPERATOR AND USED TO LOCATE ERRORS IN SOFTWARE. (THE TOOLS MAY INCLUDE DUMP, SNAP, INSPECT AND CHANGE, AND TIME CAPABILITIES.) (DAN 1201)

DECISION TABLE
A TABULAR REPRESENTATION OF THE FOLLOWING THREE ITEMS: 1. CONDITIONS- FACTORS TO CONSIDER IN MAKING A DECISION. 2. ACTIONS - STEPS TO BE TAKEN WHEN A CERTAIN COMBINATION OF CONDITIONS EXIST. 3. RULES - SPECIFIC COMBINATIONS OF CONDITIONS AND THE ACTIONS TO BE TAKEN UNDER THOSE CONDITIONS. (DAN 813) (2) A TABLE OF ALL OR SELECTED CONTINGENCIES TO BE CONSIDERED IN THE DESCRIPTION OF A PROBLEM OR THE SPECIFICATION OF A SOLUTION, TOGETHER WITH ACTIONS TO BE TAKEN IN EACH COMBINATION OF CONTINGENCIES. ALSO CALLED DECISION LOGIC TABLES." (DAN 1153)

DEFAULT
(1) AN ALTERNATIVE VALUE, ATTRIBUTE , OR OPTION THAT IS ASSUMED WHEN NONE HAS BEEN SPECIFIED AND ONE IS REQUIRED. (ANSI-X3H1) (2) TO MAKE AN ASSIGNMENT OF A VALUE, ATTRIBUTE, OR OPTION IN THE ABSENCE OF AN OTHERWISE SPECIFIED VALUE, ATTRIBUTE OR OPTION WHEN ONE IS REQUIRED. (ANSI-X3H1)

DEFICIENCY
A MISSING FUNCTION OR CAPABILITY REQUIRED IN SOFTWARE. (DAN 1201)

DELIVERABLE CODE INDICATOR
INDICATES WHETHER OR NOT THE CODE WILL BE DELIVERED TO THE CUSTOMER. (DAN 137)

DELIVERY
THE POINT AT WHICH THE SOFTWARE PACKAGE IS TURNED OVER TO A CUSTOMER FOR USE IN THE OPERATIONAL ENVIRONMENT. (DAN 21)

DEPENDENT FUNCTION
IN SOFTWARE, A PROGRAM WHICH DEPENDS ON OTHER PROGRAMS OR SUBPROGRAMS TO IMPLEMENT SOME OF ITS FUNCTIONS. (DAN 1201)

DEQUE
A DATA STRUCTURE (DOUBLE-ENDED QUEUE, PRONOUNCED "DECK") THAT, TOGETHER WITH ITS ACCESS FUNCTIONS, MODEL OPERATIONS WITH A LINEAR LIST IN WHICH INSERTIONS CAN BE MADE AT EITHER END OF THE LIST. (DAN 1153)

DESIGN
DESCRIPTION OF WHAT THE SYSTEM MUST DO, ITS COMPONENTS, THE INTERFACES AMONG

THOSE COMPONENTS, AND THE SYSTEM'S INTERFACE(S) TO THE EXTERNAL ENVIRONMENT. (SEL) (2) A DESCRIPTION OF HOW SOFTWARE WILL BE PRODUCED TO SATISFY THE SOFTWARE SPECIFICATION. (DAN 1201) (3) THAT ACTIVITY WHICH DEFINES PROGRAM DATA STRUCTURES AND LOGICAL ALGORITHMS IN RESPONSE TO, AND CONFORMING WITH, THE SOFTWARE FUNCTIONAL SPECIFICATION. IT CONSISTS OF DESCRIBING THE ORGANIZATION, DATA MANIPULATIONS, I/O PROCEDURES AND FORMATS, ETC., CARRIED TO A LEVEL OF DETAIL SUFFICIENT FOR CODING AND OPERATIONAL IMPLEMENTATION. ALSO, THE WORD "DESIGN" MAY REFER TO THE STRUCTURE OF THE PROGRAM RESULTING FROM THE DESIGN ACTIVITY, AND, THEREFORE, TO THE SOFTWARE PROGRAMMING SPECIFICATION. DEVELOPMENT OF THE SOFTWARE FUNCTIONAL SPECIFICATIONS IS SOMETIMES CALLED FUNCTIONAL DESIGN. (DAN 1153)

DESIGN ANALYSIS
DESIGN ANALYSIS ENSURES THAT THE COMPUTER PROGRAM DESIGN IS CORRECT AND THAT IT SATISFIES THE DEFINED SOFTWARE REQUIREMENTS WITH RESPECT TO DESIGN COMPLETENESS AND THE VARIOUS DESIGN ELEMENTS: MATHEMATICAL EQUATIONS, ALGORITHMS, AND CONTROL LOGIC. (DAN 306)

DESIGN ANALYZER
A SOFTWARE DESIGN TOOL WHICH CAN BE USED TO OBTAIN INFORMATION ABOUT A PROGRAM'S DESIGN AND STRUCTURE. MOST DESIGN ANALYZERS WILL ANALYZE THE CONTROL STRUCTURE AND DATA STRUCTURES OF A PROGRAM AND OUTPUT, USUALLY IN TABULAR FORM, AN INDEXED LIST OF ALL MODULES IN A PROGRAM, STATISTICS ABOUT THE MODULE, A LIST OF OTHER MODULES CALLED BY IT, AND AN INDEXED LIST OF ALL DATA BLOCKS THAT ARE ACCESSED. DESIGN ANALYZERS MAY ALSO BE USED TO PRODUCE A GRAPHICAL DESCRIPTION OF A PROGRAM'S CONTROL STRUCTURE, FOR DEBUGGING, FOR STANDARDS ENFORCEMENT, AND FOR COLLECTING RUN-TIME STATISTICS.

DESIGN HIERARCHY
IN SOFTWARE, A PROGRAM DESIGN IN WHICH THE IDENTIFIED PROGRAMMABLE ELEMENTS ARE ARRANGED IN ORDER OF DEPENDENCY FROM THE MOST DEPENDENT ELEMENTS TO THE LEAST DEPENDENT ELEMENTS. (DAN 1201)

DESIGN IMPLEMENTATION SPECIFICATIONS
THIS DOCUMENT DEFINES THE SYSTEM AND PROGRAM DESIGN. IT INCLUDES BLOCK AND PROGRAM FLOW DIAGRAMS, SET-USED INFORMATION ON ALL DATA DESCRIPTIONS OF ALL DATA AND MAY INCLUDE NARRATIVE DESCRIPTIONS OF THE OPERATION OF EVERY PROGRAM IN THE SYSTEM. (DAN LD7)

DESIGN INSPECTION
VISUAL INSPECTION OF THE DESIGN BY PERSONS OTHER THAN THE CREATOR OF THE DESIGN. (SEL)

DESIGN LANGUAGES
A COMPUTER PROGRAM USED TO PROVIDE AN UNDERSTANDABLE REPRESENTATION OF THE SOFTWARE DESIGN AS IT EVOLVES. THESE PROGRAMS ALLOW DESIGNS TO BE CONSTRUCTED AND EXPANDED IN A HIERARCHICAL FASHION. THEY DOCUMENT THE DESIGN AND THE DECISIONS THAT LED TO IT. (DAN 134)

DESIGN METHODOLOGIES
A COLLECTION OF WORK PROCEDURES TO BE USED BY THE DESIGNER TO CARRY OUT THE DESIGN TASKS REQUIRED, BY THE SYSTEM TO BE DESIGNED. (DAN 254)

DESIGN OBJECT

A CONSTRUCT USED IN SOFTWARE DESIGN, PRODUCTION AND TEST TO IDENTIFY THE LOCATION OF EACH CODED OBJECT IN ACCORDANCE WITH A LIST OF RULES. (DAN 1201)

DESIGN OBJECT NETWORK
A DRAWING DEPICTING THE HIERARCHY AND REAL INTERCONNECTIONS OF DESIGN OBJECTS WITH A SUBPROGRAM. (DAN 1201)

DESIGN OBJECTIVE
GOALS TO BE ACCOMPLISHED BY THE SOFTWARE BEING GENERATED.

DESIGN PHASE
THE CREATION AND RECORDING OF THE DESIGN, INCLUDING DISCUSSION ABOUT STRATEGY WITH PEERS. THIS PHASE DOES NOT INCLUDE THE DEVELOPMENT OF ANY CODE AT THE PROGRAMMING LANGUAGE LEVEL. IT DOES INCLUDE THE CREATION OF SPECIFICATIONS FOR SUBCOMPONENTS OF THE CURRENT COMPONENT. (SEL) (2) DURING THE DESIGN PHASE, THE SOFTWARE COMPONENT DEFINITIONS, INTERFACE, AND DATA DEFINITIONS ARE GENERATED AND VERIFIED AGAINST THE REQUIREMENTS. (SET)

DESIGN REQUIREMENTS BASELINE DOCUMENT
THIS DOCUMENT IS THE BASELINE DESCRIPTION OF THE OBJECT SYSTEM: HENCE, THE FOUNDATION UPON WHICH SYSTEM DESIGN AND CONFIGURATION CONTROL PROCEED. (DAN LD7)

DESIGN REVIEW
A SCHEDULED MEETING BETWEEN CUSTOMER AND MANUFACTURER TO DETERMINE THAT A PROPOSED SOFTWARE CONFIGURATION WILL SATISFY APPROVED PERFORMANCE SPECIFICATIONS. (DAN 1201) SEE ALSO DESIGN READING

DESIGN SIMULATION
A TECHNIQUE THAT DESCRIBES A PROPOSED SYSTEM, PRODUCES A COMPUTER BASED "MODEL" OR SIMULATED SYSTEM, AND THEN EVALUATES THE EFFECT OF VARIOUS SYSTEM REQUIREMENTS AND DESIGN ALTERNATIVES. (DAN 154)

DESIGN SPECIFICATION
A DOCUMENT CONTAINING THE APPROVED DESIGN REQUIREMENTS FOR A PROGRAM. (DAN 1201)

DESIGN VALIDATION
THE EXAMINATION OR INSPECTION OF THE FUNCTIONAL REQUIREMENTS AND THE DESIGN OF A SOFTWARE SYSTEM FOR THE PURPOSE OF FINDING ERRORS. OTHER TERMS USED TO DESCRIBE THIS TECHNIQUE OR VARIATIONS OF THIS TECHNIQUE INCLUDE DESIGN REVIEW, PROJECT REVIEW, DESIGN INSPECTIONS,WALK-THROUGHS, AND INPROCESS REVIEWS. THIS TECHNIQUE IS SIMILAR TO THE TECHNIQUE EXCEPT THAT IT IS PERFORMED EARLIER IN THE SOFTWARE DEVELOPMENT CYCLE AND AT A SYSTEM FUNCTIONAL LEVEL. (DAN 154)

DESIGN VERIFICATION
THE EXAMINATION OR INSPECTION OF A SOFTWARE SPECIFICATION FOR THE PURPOSE OF FINDING DESIGN ERRORS. OTHER TERMS USED IN THE LITERATURE TO DESCRIBE THIS TECHNIQUE OR VARIATIONS OF THIS TECHNIQUE INCLUDE DESIGN REVIEW, DESIGN INSPECTION, SPECIFICATION TESTING, PAPER TESTING WALK-THROUGH, STRUCTURED WALK-THROUGH, AND PRELIMINARY DESIGN REVIEW. (DAN 154)

DESIRED BEHAVIOR SPECIFICATION

DESCRIBING THE POSSIBLE SEQUENCING AND SIMULTANEITY OF EVENT OCCURRENCES WHICH WOULD BE CONSIDERED ACCEPTABLE DURING A SYSTEM'S OPERATION. (DAN 242)

DESK CHECKING
DESK CHECKING (DC) IS A TERM COVERING THE TOTALITY OF VERIFICATION EFFORTS PERFORMED MANUALLY DURING PROGRAM CHECKOUT WITHOUT BENEFIT OF A COMPUTER OR SIMULATOR...MOST COMMONLY, DESK CHECKING REFERS TO (1) DOING ARITHMETIC CALCULATIONS TO VERIFY OUTPUT VALUE CORRECTNESS, AND (2)"PLAYING COMPUTER" (I.E., MANUALLY SIMULATING PROGRAM EXECUTION) IN ORDER TO UNDERSTAND AND VERIFY PROGRAM LOGIC AND DATA FLOW. DESK CHECKING IS AN ESSENTIAL PART OF ANY VERIFICATION PROCESS. IT USUALLY CONCENTRATES ON AREAS OF SPECIAL PROBLEMS, ESPECIALLY SUSPECTED ERRORS OR CODE INEFFICIENCIES. ALSO, SEE - CODE VERIFICATION, PEER CODE REVIEW (SET)

DETERMINISM
THE PROPERTY OF A TRANSFORMATION PROCESS THAT THE SAME OUTPUTS ARE ALWAYS PRODUCED FOR A GIVEN SET OF INPUTS. (ANSI-X3H1)

DEVELOPMENT
THE GENERATION OF SOFTWARE FROM ITS ORIGINAL CONCEPTION THROUGH ANALYSIS, CODE PRODUCTION, CHECKOUT, DOCUMENTATION, DELIVERY, AND INTEGRATION OF THE COMPLETED PROJECT. (DAN 1201) (2) THAT PROCESS BY WHICH NEW SOFTWARE COMES INTO BEING AS A PROCESS OF DESIGN, RATHER THAN BY A PROCESS OF MODIFICATION. IT INCLUDES BOTH THE ARCHITECTURAL AND IMPLEMENTATION PHASES. (DAN 1193)

DEVELOPMENT CYCLE
SEE SOFTWARE DEVELOPMENT CYCLE

DEVELOPMENT LIFE CYCLE
SEE SOFTWARE DEVELOPMENT CYCLE

DEVELOPMENT MANAGEMENT
PROCEDURES FOR MANAGING THE DEVELOPMENT PHASE OF A SYSTEM, PROGRAM, OR SOFTWARE PROJECT. (DAN 355)

DEVELOPMENT PHASE
THE DEVELOPMENT AND RECORDING OF CODE AND IN-LINE COMMENTS BASED ON THE DESIGN. THIS PHASE INCLUDES THE MODIFICATION OF CODE CAUSED BY DESIGN CHANGES, ERRORS FOUND IN TESTING, ETC.... IT DOES NOT INCLUDE ANY TIME SPENT IN ENTERING THE CODE INTO THE COMPUTER. (SEL)

DEVELOPMENT PROCESS
SEE SOFTWARE DEVELOPMENT PROCESS

DEVELOPMENT SUPPORT LIBRARIAN
ONE OF THE MEMBERS OF THE CHIEF PROGRAMMER TEAM. THE LIBRARIAN IS RESPONSIBLE FOR THE PROGRAMMING-PRODUCT LIBRARY, CONTAINING BOTH MACHINE-AND-HUMAN-READABLE MATERIAL. THIS FUNCTION HELPS TO TRANSFORM PROGRAMMING "FROM A PRIVATE ART TO PUBLIC PRACTICE." (DAN 227) PROGRAM LIBRARY.

DEVELOPMENT TEST AND EVALUATION
TEST AND EVALUATION THAT FOCUSES ON THE TECHNOLOGICAL AND ENGINEERING ASPECTS OF THE SYSTEM, OR EQUIPMENT ITEMS. (AFR80-14)

DEVELOPMENTAL TOOLS AND TECHNIQUES
INDEXING TERM. REFERS GENERALLY TO SOME TOOL OR TECHNIQUE OR TO A GROUP OF TOOLS AND/OR TECHNIQUES USED IN MORE THAN ONE PHASE OF THE SOFTWARE DEVELOPMENT CYCLE.

DIAGNOSTIC DEBUG AIDS
COMPILE AND EXECUTION TIME CHECKOUT AND DEBUG CAPABILITIES THAT HELP IDENTIFY AND ISOLATE PROGRAM ERRORS. THESE CAPABILITIES USUALLY INCLUDE COMMANDS OR DIRECTIVES SUCH AS DUMP, TRACE, MODIFY, CONTENTS, BREAKPOINT, ETC. (DAN 134)

DIAGNOSTICS
AN INDICATION AND DESCRIPTION OF A SOFTWARE DISCREPANCY AS NOTED BY A COMPUTER PROGRAM SUCH AS A COMPILER. (NASA)

DIFFICULTY
A MEASURE OF HOW "HARD" IT IS TO ACCOMPLISH A GIVEN PROJECT. DIFFICULTY EXPRESSES THE TIME RATE OF CHANGE IN MANPOWER UTILIZATION. DIFFICULTY VARIES DIRECTLY WITH SIZE OF THE PROJECT AND INVERSELY AS THE LENGTH OF DEVELOPMENT. D=K/(TD)2 WHERE D IS DIFFICULTY, K IS SIZE AND (TD)2 IS THE SQUARE OF THE DEVELOPMENT TIME. (DAN 724)

DIGITAL AIRCRAFT CONTROL
INDEXING TERM. REFERS TO THE DEVELOPMENT OR USE OF DIGITAL COMPUTING SYSTEMS (HARDWARE AND SOFTWARE) FOR REAL-TIME AIRCRAFT CONTROL.

DIGITAL WORD
A BINARY NUMBER WHOSE BIT LENGTH CORRESPONDS TO THAT TYPICAL OF MEMORY OR BASIC ARITHMETIC OPERATIONS. (NASA)

DIRECT INTERFACE
AN INTERFACE IMMEDIATELY BETWEEN TWO SOFTWARE ELEMENTS. (DAN 21)

DISCRETE PROCESS
A PROCESS IN WHICH DATA ARE EXPRESSIBLE ONLY AS DISCRETE QUANTITIES AND ONLY AT SPECIFIC POINTS IN TIME, E.G., THE SOLUTION OF A DIFFERENCE EQUATION FOR A PARTICULAR INPUT SEQUENCE. (NASA)

DISCRETE SYSTEMS ANALYSIS
THE PROCESS OF APPLYING THE HIGHLY DEVELOPED ELECTRICAL ENGINEERING TECHNIQUES OF ANALYZING DISCRETE SYSTEMS OF TWO TERMINAL ELEMENTS TO ANALYZING THE COMPLEXITY AND EXECUTION TIME OF COMPUTER PROGRAMS. (DAN 719)

DISTINCTNESS
SOFTWARE DISTINCTNESS IS A MEASURE OF THE FAILURE-POINT INDEPENDENCE OF A PIECE OF SOFTWARE WHICH IS PERFORMING THE SAME FUNCTION AS ANOTHER PIECE OF SOFTWARE. IT IS ANALOGOUS TO HARDWARE DISTINCTNESS WHICH IS UTILIZED IN DUAL HARDWARE SYSTEMS, WHICH PERFORM THE SAME TASK AND RARELY FAIL AT THE SAME INSTANT. (DAN 781)

DISTRIBUTED PROCESSING SYSTEM
A COOPERATIVE DISTRIBUTED PROCESSING SYSTEM IS DEFINED AS A COLLECTION OF INTERCONNECTED PROCESSING ELEMENTS WITH DECENTRALIZED CONTROL THAT PERMITS COOPERATION AMONG PROCESSORS FOR THE EXECUTION OF A SINGLE TASK. (DAN 273)

DISTRIBUTED SYSTEMS ARE AN APPROPRIATE RESPONSE TO DISTRIBUTED FUNCTIONS TO BE PERFORMED. THE FUNCTIONS MAY BE DISTRIBUTED GEOGRAPHICALLY , OPERATIONALLY OR MANAGERIALLY. THE IMPORTANT CHARACTERISTIC IS THAT THEY BE FUNCTIONALLY INDEPENDENT OF ONE ANOTHER AND HAVE WEAK, WELL-DEFINED DATA FLOW ORIENTED INTERACTIONS. (DAN 346) (3) A COOPERATIVE ARRANGEMENT OF INTERCONNECTED COMPUTERS WHOSE QUASI-AUTONOMOUS OPERATIONS ARE COORDINATED BY A REASSIGNABLE EXECUTIVE PROGRAM. (NASA)

DOCUMENTATION
SOFTWARE DOCUMENTATION IS TECHNICAL DATA, INCLUDING COMPUTER LISTINGS AND PRINTOUTS, IN HUMAN-READABLE FORM WHICH (1) DOCUMENTS THE DESIGN OR DETAILS OF THE SOFTWARE, (2) EXPLAINS THE CAPABILITIES OF THE SOFTWARE, OR (3) PROVIDES OPERATING INSTRUCTIONS FOR USING THE SOFTWARE TO OBTAIN DESIRED RESULTS FROM COMPUTER EQUIPMENT. (2) WRITTEN MATERIAL, OTHER THAN SOURCE CODE STATEMENTS, THAT DESCRIBES A SYSTEM OR ANY OF ITS COMPONENTS. (SEL) (3) THE PRODUCTION OF ALL THE PAPER WORK NECESSARY TO DESCRIBE THE FINAL PRODUCT. EXAMPLES INCLUDE: CROSS-REFERENCE LISTINGS, DICTIONARY LISTINGS, AND FLOW CHARTS. (DAN LD7) (4) THE COMPREHENSIVE DESCRIPTION OF A COMPUTER PROGRAM IN VARIOUS FORMATS AND LEVELS OF DETAIL TO CLEARLY DEFINE ITS CONTENT AND COMPOSITION. (NASA)

DOCUMENTATION GENERATOR
A COMPUTER PROGRAM USED TO SHOW IN DETAIL THE LOGICAL STRUCTURE OF A COMPUTER PROGRAM, USUALLY BY PRODUCING FLOWCHARTS.

DOCUMENTATION LEVEL
SPECIFICATION OF THE DEGREE OF DETAIL AND THE FORMAT QUALITY FOR A PARTICULAR ITEM OF DOCUMENTATION. (DAN 1153)

DOD COMMON HIGH ORDER LANGUAGE
AN EFFORT TO CREATE A SITUATION IN WHICH SOFTWARE FOR NEW EMBEDDED COMPUTER SYSTEMS ARE DEVELOPED AND MAINTAINED USING A MINIMAL NUMBER OF GENERAL-PURPOSE PROGRAMMING LANGUAGES, AND THAT THOSE LANGUAGES BE SUITED TO THE APPLICATIONS, BE WIDELY USED IN DOD, AND BE WELL SUPPORTED. (DAN 251)

DOMAIN
COMPLETE SET OF ALLOWABLE VALUES OR ITEMS.

DOMAIN ERROR/FAULT
AN ERROR/FAULT IN THE CONTROL FLOW OF THE PROGRAM WHICH CAUSES A SPECIFIC INPUT TO FOLLOW THE WRONG PATH. (DAN 842)

DREAM
(DESIGN REALIZATION, EVALUATING AND MODELING SYSTEM). AN INTERACTIVE DESIGN TOOL. (DAN 242)

DRIVER PROGRAM
SUPERFLUOUS (THROW-AWAY) CODE NEEDED TO PERFORM THE UNIT TESTING AND LOWER LEVELS OF INTEGRATION TESTING IN A BOTTOM UP SOFTWARE DEVELOPMENT EFFORT. (DAN 154)

DRIVERS
COMPUTER BASED INFORMAL TESTING TECHNIQUES WHICH ARE USED IN CONJUNCTION WITH OTHER TECHNIQUES. DRIVERS ARE USED ALMOST EXCLUSIVELY DURING THE SYSTEM

IMPLEMENTATION PHASE OF A SOFTWARE DEVELOPMENT PROJECT. (DAN 154)

DUAL CODE
SOURCE CODE WRITTEN IN TWO VERSIONS BY DIFFERENT PROGRAMMERS OR DIFFERENT PROGRAMMING TEAMS. THE SOURCE CODE MAY BE IN THE SAME OR DIFFERENT LANGUAGES. THE PURPOSE MAY BE TO PROVIDE AN ERROR DETECTION OR DEBUGGING TOOL, INCREASE RELIABILITY, PROVIDE ADDITIONAL HIGH LEVEL DOCUMENTATION, OR TO REDUCE THE PROBABILITY OF SYSTEMATIC PROGRAMMING CODE ERRORS OR COMPILER ERRORS INFLUENCING THE END RESULT. (DAN 781)

DUMP
DATA THAT HAVE BEEN DUMPED. (ANSI-X3) (2) TO WRITE THE CONTENTS OF A STORAGE, OR OF PART OF A STORAGE, USUALLY FROM AN INTERNAL STORAGE TO AN EXTERNAL MEDIUM, FOR A SPECIFIC PURPOSE SUCH AS TO ALLOW OTHER USE OF THE STORAGE, AS A SAFEGUARD AGAINST FAULTS OR ERRORS, OR IN CONNECTION WITH DEBUGGING. (ANSI-X3) (3) A RECORD OF THE STATE OF THE MEMORY SPACE USED BY A PROGRAM AT SOME POINT IN ITS EXECUTION. A DUMP MAY INCLUDE ALL OR PART OF THE PROGRAM'S MEMORY SPACE (INCLUDING REGISTERS). (SEL)

DYNAMIC ALLOCATION
THE ALLOCATION OF CORE SPACE FOR MEMORY REQUIRED BY AN OPERATING PROGRAM DURING ITS EXECUTION PHASE

DYNAMIC REDUNDANCY
DYNAMIC REDUNDANCY TECHNIQUE IS USED IN CONJUNCTION WITH STATIC REDUNDANCY, SOFTWARE SUPPORT AND HUMAN ASSISTANCE FOR FAULT DETECTION. RECOVERY ACTION IS ACCOMPLISHED BY SWITCHING OVER TO THE STANDBY UNIT. (DAN 311)

DYNAMIC RESTRUCTURING
CHANGING SOFTWARE SYSTEM PARTS WHILE THE SYSTEM IS RUNNING. ALSO: DYNAMIC MODIFICATION THAT NECESSARILY IMPLIES DATA CONVERSION, EITHER ON DEMAND OR ON REQUEST. (DAN 278)

DYNAMIC SIMULATOR
A COMPUTER PROGRAM USED TO CHECK OUT A PROGRAM IN A SIMULATED ENVIRONMENT SIMILAR TO THAT IN WHICH IT WILL RESIDE. CLOSED-LOOP EFFECTS BETWEEN COMPUTER AND ENVIRONMENTAL MODELS ARE GAINED WHEN INPUTS AND OUTPUTS ARE RESPONDED TO BY THE VARIOUS MODELS. THE SIMULATOR ALLOWS THE ENVIRONMENT TO BE STABLIZED AT A SPECIFIC CONFIGURATION FOR ANY NUMBER OF RUNS REQUIRED TO OBSERVE, DIAGNOSE, AND RESOLVE PROBLEMS IN THE OPERATIONAL PROGRAM. (DAN 134)

DYNAMIC TESTING
A TESTING SYSTEM WHICH BUILDS A COMPREHENSIVE RECORD OF A SINGLE EXECUTION OF A PROGRAM. THIS RECORD - THE EXECUTION HISTORY- IS USUALLY OBTAINED BY INSTRUMENTING THE SOURCE PROGRAM WITH MONITORING CODE WHOSE PURPOSE IS TO CAPTURE INFORMATION ABOUT THE PROGRESS OF THE EXECUTION. THE EXECUTION HISTORY IS SAVED IN A FILE, USUALLY IN TABULAR OR STATISTICAL FORM, SO THAT AFTER THE EXECUTION TERMINATES, THE EXECUTION HISTORY CAN BE PERUSED BY THE TESTER. (DAN 360 - MODIFIED) TESTING A PROGRAM OR SUBPROGRAM AS PART OF A SOFTWARE SYSTEM WHILE SUBJECTING IT TO A REAL OR SIMULATED ENVIRONMENT. (DAN 1201)

EDITOR

A COMPUTER PROGRAM USED TO ANALYZE SOURCE PROGRAMS FOR CODING ERRORS AND TO EXTRACT INFORMATION THAN CAN BE USED FOR CHECKING RELATIONSHIPS BETWEEN SECTIONS OF CODE. THE EDITOR WILL SCAN SOURCE CODE AND DETECT VIOLATIONS TO SPECIFIC PROGRAMMING PRACTICES AND STANDARDS, CONSTRUCT AN EXTENSIVE CROSS-REFERENCE LIST OF ALL LABELS, VARIABLES, AND CONSTANTS, AND CHECK FOR PRESCRIBED PROGRAM FORMATS. (DAN 134) (2) A COMPUTER PROGRAM WHICH PERMITS SELECTIVE REVISION OF A SOURCE PROGRAM. (NASA)

EDUCATION

A DISCIPLINE AND DEVELOPMENT BY MEANS OF STUDY AND LEARNING. IN THIS GLOSSARY APPLIED STRICTLY TO SOFTWARE AND SOFTWARE ENGINEERING; REFERS TO FORMAL AND INFORMAL EDUCATIONAL PROCESSES,INCLUDING TECHNOLOGY TRANSFER.

EFFECTIVENESS

SYSTEM READINESS, AND DESIGN ADEQUACY. EFFECTIVENESS IS EXPRESSED AS THE PROBABILITY THAT THE SYSTEM CAN SUCCESSFULLY MEET AN OPERATIONAL DEMAND WITHIN A GIVEN TIME WHEN OPERATED UNDER SPECIFIED CONDITIONS. (DAN781)

EFFICIENCY

CODE POSSESSES THE CHARACTERISTIC EFFICIENCY TO THE EXTENT THAT IT FULFILLS ITS PURPOSE WITHOUT WASTE OF RESOURCES. THIS IMPLIES THAT CHOICES OF SOURCE CODE CONSTRUCTIONS ARE MADE IN ORDER TO PRODUCE THE MINIMUM NUMBER OF WORDS OF OBJECT CODE, OR THAT WHERE ALTERNATE ALGORITHMS ARE AVAILABLE, THOSE TAKING THE LEAST TIME ARE CHOSEN; OR THAT INFORMATION-PACKING DENSITY IN CORE IS HIGH, ETC. OF COURSE, MANY OF THE WAYS OF CODING EFFICIENTLY ARE NOT NECESSARILY EFFICIENT IN THE SENSE OF BEING COST-EFFECTIVE, SINCE PORTABILITY, MAINTAINABILITY, ETC., MAY BE DEGRADED AS A RESULT. (DAN 239) THE PROCESS WHOSE END IS TO INCREASE EFFICIENCY IS OPTIMIZATION. EFFICIENCY IS THE RATIO OF USEFUL WORK PERFORMED TO THE TOTAL ENERGY EXPENDED. IT CAN ALSO BE EXPRESSED AS THE EFFECTIVENESS/COST RATIO. (DAN 781)

EGOLESS PROGRAMMING

EGOLESS PROGRAMMING IS THE PROCESS OF PROGRAMMING WHICH AVOIDS INVOLVING PSYCHOLOGICAL MECHANISMS THAT SERVE TO PROTECT THE PROGRAMMER'S SELF IMAGE OR SELF ESTEEM...THIS TERM WAS COINED BY WEINBERG (THE PSYCHOLOGY OF COMPUTER PROGRAMMING, VAN NOSTRAND REINHOLD COMPANY, NEW YORK 1971). WEINBERG FELT THAT SOME PROGRAMMERS CAN BECOME PSYCHOLOGICALLY ATTACHED TO THEIR PROGRAMS AS EXTENSIONS OF THEMSELVES SO THAT ERRORS IN PROGRAMS BECOME DAMAGING TO THE PROGRAMMER'S SELF-IMAGE. AN EGOLESS PROGRAMMING ENVIRONMENT IS ONE THAT CREATES AN ATTITUDE THAT OPEN, SHARED PROGRAMMING IS GOOD. BY MAKING CODE PUBLICLY AVAILABLE, THE OWNERHIP OF CODE IS DISCOURAGED, AND INDIVIDUALS ARE ENCOURAGED TO WRITE CODE THAT WILL BE CLEAR AND UNDERSTANDABLE TO OTHERS. (SET)

ELEMENT

A GROUPING OF ROUTINES WHICH PERFORMS A PRESCRIBED FUNCTION. (DAN 21)

EMBEDDED COMPUTER SYSTEMS

AN EMBEDDED COMPUTER SYSTEM IS A COMPUTER SYSTEM THAT IS INTEGRAL TO AN ELECTROMECHANICAL SYSTEM SUCH AS A WEAPON, AIRCRAFT, SHIP, MISSILE, SPACECRAFT, COMMAND AND CONTROL, RAPID TRANSIT SYSTEM, AND THE LIKE... EMBEDDED COMPUTER SYSTEMS ARE CONSIDERED DIFFERENT THAN AUTOMATIC DATA PROCESSING SYSTEMS (ADPS),PRIMARILY IN THE CONTEXT OF HOW THEY ARE DEVELOPED, ACQUIRED, AND OPERATED IN A USING SYSTEM. THE KEY ATTRIBUTES OF

AN EMBEDDED COMPUTER SYSTEM ARE: (A) IT IS A COMPUTER SYSTEM THAT IS PHYSICALLY INCORPORATED INTO A LARGER SYSTEM WHOSE PRIMARY FUNCTION IS NOT DATA PROCESSING. (B) IT IS INTEGRAL TO SUCH A LARGER SYSTEM FROM A DESIGN, PROCUREMENT, OR OPERATIONS VIEWPOINT. (C) ITS OUPUTS GENERALLY INCLUDE INFORMATION, COMPUTER CONTROL SIGNALS, AND COMPUTER DATA. (SET)

EMBEDDED LANGUAGES
EMBEDDING IS THE (SEMANTIC) EXTENSION OF A PROGRAMMING LANGUAGE WITHOUT ALTERING EITHER THE EXISTING FACILITIES OF THAT LANGUAGE OR ITS PROCESSOR, PREFERABLY WRITING THE EXTENSION IN THE LANGUAGE ITSELF. (AN EMBEDDED EXTENSION ADDS SEMANTIC CAPABILITY BY MAKING IT EASIER FOR A USER TO DO SOMETHING WHICH WAS POSSIBLE ALREADY IN THE EXISTING LANGUAGE. I.E. A SUBROUTINE) (DAN 448)

EMBEDDED SOFTWARE
SOFTWARE RESIDENT IN A SYSTEM DEDICATED TO A FUNCTION OTHER THAN THAT OF DIGITAL COMPUTATION IN GENERAL. (NASA)

EMULATION
THE PROCESS OF IMITATING ONE SYSTEM WITH ANOTHER SO THAT THE IMITATING SYSTEM ACCEPTS THE SAME DATA, EXECUTES THE SAME COMPUTER PROGRAMS, AND ACHIEVES THE SAME RESULTS AS THE IMITATED SYSTEM. EMULATION IS OFTEN USED AS A DEVELOPMENTAL TECHNIQUE AND IS ALSO USED AS AN APPLICATION IN WEAPONS SYSTEMS, LOGISTICS, ETC. (DAN 300) (2) THE USE OF A HOST COMPUTER TO EXECUTE THE MACHINE LEVEL INSTRUCTIONS OF A TARGET COMPUTER IN THE SAME MANNER AS THE TARGET COMPUTER ITSELF WOULD, AS WITH RESPECT TO DETAILS SUCH AS WORD LENGTH, OVERFLOW, AND TIME-SCALED OPERATION. (NASA)

ENCAPSULATION
THE PROCESS OF ISOLATING THE CHANGEABLE PARTS OF A PROGRAM IN MODULES.

ENCODING
A DESIGN AND PROGRAMMING TECHNIQUE THROUGH WHICH INFORMATION IS STORED OR CONVEYED BY MEANS OF A REVERSIBLE MAPPING FROM THE DOMAIN IN WHICH THE INFORMATION EXISTS ORIGINALLY INTO ANOTHER DOMAIN. (ABBOTT)

END DATE
DATE WHEN PROJECT IS SCHEDULED TO BE COMPLETED. (SEL)

ENGINEERING
APPLIED SCIENCE CONCERNED WITH THE UTILIZATION OF RAW MATERIALS, PRODUCTS OF TECHNOLOGY, AND PHYSICAL LAWS FOR SUPPLYING HUMAN NEEDS. A PROFESSION CHARACTERIZED BY THE PROPENSITY TO SOLVE TECHNOLOGICAL AND RELATED PROBLEMS WITH GIVEN CONSTRAINTS IN AN ORGANIZED, RESPONSIBLE WAY. (DAN 1153)

ENGINEERING CHANGE PROPOSAL
THE FORMAL VEHICLE BY WHICH A CHANGE IN THE BASELINE DOCUMENT IS PROPOSED. IT DESCRIBES THE NATURE AND MAGNITUDE OF A PROPOSED CHANGE AND THE IMPACT OF THE CHANGE ON ALL ELEMENTS OF THE SYSTEM. (DAN LD7)

ENGINEERING SCIENTIFIC SIMULATIONS
AN ENGINEERING SIMULATION IS USED TO STUDY SYSTEM CHARACTERISTICS, DEVELOP ALGORITHMS, AND PROVIDE DATA THAT ACT AS A STANDARD FOR TESTING. THESE PROGRAMS GENERALLY SIMULATE SUBSYSTEMS AT VARYING DEGREES OF COMPLEXITY

DEPENDING ON THE SUBSYSTEM BEING STUDIED OR THE USE MADE OF THE SIMULATION. THEY GENERALLY CONSIST OF A SET OF MODULES, EACH OF WHICH IS ASSIGNED A SPECIFIC SIMULATION FUNCTION AND IS DESIGNED WITH WELL-DEFINED INPUTS AND OUTPUTS AND PRECISE INTERFACES. EACH MODULE PERFORMS AN ASSIGNED SIMULATION FUNCTION TO VARY THE METHOD, SPEED OF COMPUTATION, ACCURACY, COMPLEXITY, ETC. THE STRUCTURE CAN ENCOMPASS ALL BASIC SIMULATION CAPABILITIES FOR SIMULATION OF CONTINUOUS AND DISCRETE SYSTEMS. (DAN 134)

ENGLISH (OR INFORMAL) SPECIFICATIONS

SPECIFICATIONS GIVEN AS READABLE ENGLISH TEXT, AS OPPOSED TO SOME FORMAL NOTATION. (SEL)

ENTRY

ENTRY IS THE INSTRUCTION AT WHICH THE EXECUTION OF A ROUTINE BEGINS... A "PROPER PROGRAM" IS ONE HAVING ONLY ONE ENTRY AND ONE EXIT. ADDITIONAL ENTRIES IMPLY INCREASED COMPLEXITY BOTH IN COUPLING AND IN INTERNAL FUNCTIONAL COMPOSITION. MULTIPLE ENTRIES ARE PROHIBITED IN STRUCTURED PROGRAMMING GUIDELINES. (SET)

ENVIRONMENT

THE COMBINATION OF ALL EXTERNAL OR EXTRINSIC CONDITIONS THAT AFFECT THE OPERATION OF AN ENTITY. (ANSI-X3H1)

ENVIRONMENTAL SIMULATOR

ENVIRONMENTAL SIMULATORS ARE SOFTWARE AND/OR HARDWARE TEST TOOLS WHICH PROVIDE REALISTIC INPUT SIGNALS AT EQUIPMENT INTERACES AND RESPOND, IN A DYNAMIC MANNER, TO OUPUT SIGNALS GENERATED BY THE SOFTWARE OR EQUIPMENT UNDER TEST... AN ENVIRONMENTAL SIMULATOR SHOULD BE DESIGNED SO THAT IT SIMULATES THE RUN-TIME ENVIRONMENT OF THE TEST ARTICLE SOFTWARE. ENVIRONMENTAL SIMULATORS MAY BE ANALOG, DIGITAL, OR HYBRID, DEPENDING ON THE TYPE AND DEGREE OF TESTING TO BE ACCOMPLISHED. TEST ARTICLE SOFTWARE MAY BE TEMPORARILY SPECIALLY INSTRUMENTED TO PROVIDE ADDITIONAL TEST DATA WHEN RUN WITH AN ENVIRONMENTAL SIMULATOR. (SET) (2) A COMPUTER PROGRAM USED TO PERMIT TESTING OF OPERATIONAL PROGRAMS ON A HOST COMPUTER. THE OPERATIONAL PROGRAMS RUN UNDER SIMULATED CONDITIONS AS IF THEY WERE OPERATING WITHIN THE REAL-TIME CONTROL PROGRAM OF A MACHINE TO WHICH ALL OF THE DEVICES CONSTITUTING THE ULTIMATE SYSTEM ARE ATTACHED. THE SIMULATOR PROGRAM CONTAINS EXPANSIONS OF ALL CONTROL PROGRAM MACROS THAT MODIFY THE ENTRY BLOCK AND OTHER WORKING STORAGE IN THE SAME MANNER AS THE MACROS IN THE ACTUAL CONTROL PROGRAMS. (DAN 134)

EQUATE PROGRAM

A COMPUTER PROGRAM THAT LISTS ALL EQUIVALENCES FOUND IN EXAMINED CODE AND PRINTS WARNINGS WHEN MULTIPLE EQUIVALENCES ARE FOUND. (DAN 134)

EQUIVALENCE OF MONITOR IMPLEMENTATIONS

THE PROCESS OF ANALYZING MONITOR IMPLEMENTATION CONVENTIONS BY FORMAL LOGICAL PROOFS, MAKING RIGOROUS COMPARISONS TO PROVE THAT ONE CONVENTION MAY OR MAY NOT BE SUBSTITUTED FOR ANOTHER WHILE PRESERVING THE CLASS OF PROVABLE PROGRAM PROPERTIES. OR- 2 OR MORE MONITOR IMPLEMENTATIONS ARE EQUIVALENT IF ONE CONVENTION MAY BE SUBSTITUTED FOR ANOTHER WHILE PRESERVING THE CLASS OF PROVABLE PROGRAM PROPERTIES. (DAN 421)

ERROR

AN ERROR IS A DISCREPANCY WHICH RESULTS IN SOFTWARE CONTAINING A FAULT. (2) AN ERROR IS AN ACTION WHICH RESULTS IN SOFTWARE CONTAINING A FAULT. THE ACT OF MAKING AN ERROR INCLUDES OMISSION OR MISINTERPRETATION OF USER REQUIREMENTS IN THE SOFTWARE SUB-SYSTEM SPECIFICATION, INCORRECT TRANSLATION OR OMISSION OF A REQUIREMENT IN THE DESIGN SPECIFICATION AND PROGRAMMING ERRORS. ALSO, PROGRAMMING ERRORS INCLUDE: ALGORITHMIC (FAILS PROOF OF CORRECTNESS), ALGORITHMIC APPROXIMATION (ACCURATE FOR SOME INPUTS, INACCURATE FOR OTHERS), TYPOGRAPHICAL (E.G., I FOR 1,* FOR **, ETC.), DATA STRUCTURE (E.G., DIMENSIONS, LINKAGES INCORRECT), SEMANTIC (COMPILER WORKS DIFFERENTLY THAN PROGRAMMER BELIEVES), SYNTAX (E.G., PARENTHESES OMITTED), LOGIC (E.G., OR FOR XOR), INTERFACE (I/O MISMATCH), TIMING (E.G., EXECUTION TIME OF INSTRUCTION SEQUENCE GREATER THAN REQUIRED). (SET) (3) A DISCREPANCY BETWEEN A SPECIFICATION AND ITS IMPLEMENTATION. THE SPECIFICATION MIGHT BE REQUIREMENTS, DESIGN SPECIFICATIONS, CODING SPECIFICATIONS, ETC. (SEL) (4) (ISO) A DISCREPANCY BETWEEN A COMPUTED, OBSERVED, OR MEASURED VALUE OR CONDITION AND THE TRUE, SPECIFIED, OR THEORETICALLY CORRECT VALUE OR CONDITION. (ANSI-X3)

ERROR ANALYSIS

ANALYSIS OF ERRORS WITH THE PURPOSE OF TRACING ERRORS TO THEIR SOURCES IN BOTH LIFE CYCLE PHASE AND IN CONDITIONS WHICH RESULT IN ERRORS BEING MADE.

ERROR CATEGORIES

SOFTWARE ERRORS INCLUDING THE FOLLOWING: COMPUTATIONAL ERRORS, LOGIC ERRORS, DATA INPUT ERRORS, DATA HANDLING ERRORS, DATA OUTPUT ERRORS, INTERFACE ERRORS, ARRAY PROCESSING ERRORS, DATA BASE ERRORS, OPERATION ERRORS, PROGRAM EXECUTION ERRORS, DOCUMENTATION ERRORS. (DAN 226)

ERROR CATEGORIZATION

THE PROCESS OF SELECTING CATEGORIES AND ASSIGNING VARIOUS ERRORS TO THOSE CATEGORIES; REFERS ESPECIALLY TO METHODS OF SELECTING CATEGORIES. (FULL LISTING OF CATEGORIES AND SUBCATEGORIES, P45-50 (DAN 295))

ERROR CONFINEMENT

HARDWARE MECHANISM(S) WHICH DO NOT PREVENT OCCURRENCES OF ERRORS BUT DO LIMIT THEIR INCIDENCE ON THE PROTECTED ENTITIES OF A SYSTEM. (DAN 209)

ERROR CORRECTION

THE PROCESS OF CORRECTING A SOFTWARE ERROR. THE CORRECTIVE ACTION MAY BE TAKEN BY THE PROGRAMMER, (PHASE 1) OR THE CORRECTIVE ACTION MAY BE TAKEN BY A SYSTEM ANALYST OR SYSTEM DESIGNER (PHASE 2) (DAN 296)

ERROR CORRECTION COSTS

THE COST OF CORRECTING ERRORS IN SOFTWARE. THE COST IS DIRECTLY RELATED TO THE PHASE IN WHICH THE ERROR IS DISCOVERED; THE LATER IN THE DEVELOPMENT CYCLE, THE HIGHER THE COST IS LIKELY TO BE.

ERROR CORRECTION LIMIT POLICIES

POLICIES DESIGNED TO DETERMINE THE OPTIMUM TIME VALUE THAT MINIMIZE THE LONG RUN AVERAGE COST OF DEBUGGING AT 2 LEVELS - CORRECTIVE ACTION UNDERTAKEN BY THE PROGRAMMER (PHASE 1); AND ACTION UNDERTAKEN BY A SYSTEM ANALYST OR SYSTEM DESIGNER (PHASE 2), IF THE ERROR IS NOT CORRECTED IN PHASE (1). (DAN 298)

ERROR DATA
INDEXING TERM. DATA RELATING TO SOFTWARE ERRORS; DATA MAY INCLUDE A DESCRIPTION OF TYPE OF ERROR, WHERE IN THE SOFTWARE DEVELOPMENT CYCLE THE ERROR WAS MADE, WHEN DISCOVERED, WHEN CORRECTED, AND THE SOLUTION.

ERROR DATA ACQUISITION
THE PROCESS AND/OR METHODS OF OBTAINING SOFTWARE ERROR DATA. COMMON VEHICLES ARE REPORTING FORMS, INTERVIEWS, AUTOMATIC COLLECTION OF DATA BY THE COMPUTING SYSTEM, OR BY USE OF AUTOMATIC DATA ANALYSIS ROUTINES.

ERROR DATA ANALYSIS
ANALYSIS OF ERROR DATA MAY INCLUDE THE CATEGORIZATION OF ERRORS INTO TYPES, COMPUTATION OF THE MEAN TIMES TO DISCOVERY OF THE VARIOUS TYPES OF ERRORS, THE PERCENTAGE OF ERRORS FALLING INTO EACH CATEGORY, THE PERCENTAGE OF ERRORS TRACEABLE TO EACH STEP OF THE DEVELOPMENT PHASE OF THE SOFTWARE LIFE CYCLE, BY TYPE AS WELL AS BY NUMBER, ERROR RATES RELATED TO FUNCTIONAL AREA OF THE SOFTWARE, AND MEAN TIME TO CORRECT THE VARIOUS TYPES OF ERRORS AND/OR COST OF CORRECTING ERRORS. ANALYSIS MAY BE DONE MANUALLY OR BY USE OF AUTOMATED DATA ANALYSIS ROUTINES.

ERROR DETECTION
THE PROCESS OR TECHNIQUE, USED IN IMPLEMENTATION OF FAULT-TOLERANT SOFTWARE, OF INSERTING EXTRA CODE WHICH EXPLICITLY TESTS KEY VARIABLES FOR CORRECTNESS.

ERROR DETECTION CODE
A CODE IN WHICH EACH EXPRESSION CONFORMS TO SPECIFIC RULES OF CONSTRUCTION SO THAT IF CERTAIN ERRORS OCCUR IN AN EXPRESSION THE RESULTING EXPRESSION WILL NOT CONFORM TO THE RULES OF CONSTRUCTION AND THUS THE PRESENCE OF THE ERRORS IS DETECTED. SYNONYMOUS WITH SELF-CHECKING CODE. (ANSI-X3)

ERROR DETECTION PROCESS
THE CARRYING OUT OF THOSE ACTIVITIES WHICH DETERMINE THAT SOFTWARE CONTAINS ONE OR MORE SPECIFIC MANIFESTATIONS OF AN ERROR.

ERROR ISOLATION
TECHNIQUES USED TO IMPLEMENT FAULT-TOLERANT COMPUTING. THE BASIC STRATEGY USED IS THE ATTEMPT TO ISOLATE ERRORS TO A MINIMAL PART OF THE SOFTWARE SYSTEM SO THAT IF AN ERROR IS ENCOUNTERED, THE TOTAL SYSTEM DOES NOT BECOME INOPERABLE; EITHER ISOLATED FUNCTIONS WITHIN THE SYSTEM BECOME INOPERABLE OR PARTICULAR USERS CAN NO LONGER CONTINUE TO FUNCTION. OTHER ERROR ISOLATION TECHNIQUES ARE CONCERNED WITH PROTECTING EACH PROGRAM IN THE SYSTEM FROM ERRORS IN OTHER PROGRAMS. (DAN 286)

ERROR MANAGEMENT
AN AUTOMATED PROCEDURE FOR RELIABILITY ESTIMATION USING ANY OR ALL OF 6 BASIC PROCEDURES A) MANUAL REGISTRATION OF ERROR INFORMATION B) AUTOMATIC REGISTRATION OF ERROR INFORMATION C) STATUS REGISTRATION OF ERROR CORRECTION ACTIVITIES, D) INQUIRY FOR STATISTICAL ERROR INFORMATION OR STATUS INFORMATION OF EACH ERROR, E) AUTHORIZATION OF ERROR F) RELIABILITY ESTIMATION. (DAN 246)

ERROR MODELING
THE PROCESS OF DEVELOPING A MODEL OR MODELS FOR SOFTWARE ERROR PREDICTION OR

RELIABILITY ASSESSMENT. (DAN 296)

ERROR MODELS
MATHEMATICAL MODELS FOR PREDICTING SUCH QUANTITIES AS THE NUMBER OF REMAINING ERRORS IN A SOFTWARE PACKAGE, THE TIME TO ACHIEVE A DESIRED RELIABILITY LEVEL AND A MEASURE OF THE SOFTWARE RELIABILITY. (DAN 296)

ERROR PREDICTION
THE PROCESS OF PREDICTING SUCH QUANTITIES AS NUMBER OF ERRORS, TYPES OF ERRORS, AND TIMES TO DISCOVERY OF THE NEXT N ERRORS IN A SOFTWARE PACKAGE. (DAN 296)

ERROR PREDICTION MODELS
MATHEMATICAL MODELS FOR PREDICTING THE NUMBER OF REMAINING ERRORS IN A SOFTWARE PACKAGE. (DAN 296) SEE ALSO (DAN 239) PG 511 AND 510.

ERROR PREVENTION
THE CAUSE OF AN ERROR IS A DISCREPANCY BETWEEN THE DIFFICULTY OF THE PROBLEM AND THE ADEQUACY OF THE MEANS APPLIED. PREVENTION OF ERRORS IS THEN DEFINED AS THE APPLICATION OF ALL MEASURES CAPABLE OF REDUCING THIS DISCREPANCY. (DAN 559)

ERROR RECOVERY
THE ABILITY OF A SYSTEM TO RETURN TO A RELIABLE UP STATE AFTER A FAILURE (DAN 476)

ERROR SEVERITY
A DESCRIPTION (NUMERICAL OR VERBAL) OF AN ERROR'S EFFECT UPON THE RESULTS OF EXECUTING A PROGRAM IN COMPARISON TO THE EXPECTED RESULTS. SEE ALSO: MAJOR ERROR, MINOR ERROR

ESTIMATING
DETERMINING WHAT LEVELS OF EFFORT AND WHAT RESOURCES NEED TO BE APPLIED TO ACCOMPLISH THE DESIRED RESULTS. ALSO SEE BUDGETING AND ESTIMATING.

EUCLID
A PASCAL-BASED LANGUAGE WHICH IS USED FOR WRITING SYSTEM PROGRAMS THAT ARE TO BE VERIFIED. (DAN 389) (BY AUTOMATED MEANS)

EVENT-BASED MODEL
A MODEL WHICH DESCRIBES NON-SEQUENTIAL BEHAVIOR OR THE INTERRELATIONSHIPS BETWEEEN THE BEHAVIORS OF SEVERAL COMPONENTS OF A SOFTWARE SYSTEM IN TERMS OF SIGNIFICANT OCCURENCES DURING SYSTEM OPERATION (EVENTS). AN EVENT-BASED MODEL CAN BE USED AS A DESIGN OR SPECIFICATION TECHNIQUE.

EVOLUTION
EVOLUTION IS A DESIGNED CHARACTERISTIC OF A SYSTEM DEVELOPMENT WHICH INVOLVES GRADUAL STEPWISE CHANGE. A COMPLEX SYSTEM CAN BE IMPLEMENTED IN SMALL STEPS; EACH STEP CAN HAVE A CRITERION FOR SUCCESSFUL ACHIEVEMENT AS WELL AS A "RETREAT POSSIBILITY" TO A PREVIOUS SUCCESSFUL STEP IN THE EVENT OF FAILURE. (DAN 781-MOD) (2) COMPARE WITH BUILDS. (3) SEE ALSO IMPLEMENTATION PHASE, AND SYSTEM INTEGRATION.

EVOLUTIONARY SYSTEMS

SYSTEMS WHICH ARE DESIGNED TO BEGIN AS A MODEST SET OF FUNDAMENTAL CAPABILITIES BUT ARE ABLE TO GROW PAINLESSLY INTO MORE COMPLEX CONFIGURATIONS AS NEW MISSION REQUIREMENTS ARISE IN THE FUTURE. PRINCIPLES CONSIDERED FUNDAMENTAL: VIRTUALIZATION, INTERFACE CONTROL, INTEROPERABILITY AND AUTOMATABILITY. (DAN 346)

EXECUTE
THIS WORD IS TO BE AVOIDED (ANSI-X3H1)

EXECUTION
EXECUTION OF A PROGRAM CAUSES THE OPERATIONS IN ITS ALGORITHM TO BE PERFORMED IN SOME ORDER. THE ORDER IN WHICH AN ALGORITHM'S OPERATIONS ARE PERFORMED IS DETERMINED BY THE CONTROL STRUCTURES AND CONTROL STATEMENTS USED TO ORGANIZE THE OPERATIONS. A PROGRAM IS EXECUTED BY A PROCESSOR. (ABBOTT)

EXECUTION ANALYSIS
THE AUTOMATED MONITORING OF THE COMPUTER BASED SOFTWARE TESTING ACTIVITIES, COLLECTION OF DATA FROM THESE TESTING ACTIVITIES, AND SUBSEQUENTLY PREDICTING, BY MANUALLY ANALYZING THE DATA, THE DURATION AND COST OF TESTING AND THE QUALITY OF THE SOFTWARE PRODUCT. OTHER TERMS USED IN THE LITERATURE REFERRING TO THIS TECHNIQUE OR VARIATIONS OF THIS TECHNIQUE INCLUDE CODE ANALYZER, CODE AUDITOR, PROGRAM EVALUATOR, AND PRODUCT ASSURANCE EVALUATOR. (DAN 154)

EXECUTION TIME
THE ACTUAL PROCESSOR TIME UTILIZED IN EXECUTING A PROGRAM. SEE ALSO: ERROR DATE PARAMETERS, EXECUTION TIME THEORY, FAILURE DATA.

EXECUTION TIME THEORY
A WAY FOR THE COMPUTATION CENTER MANAGER TO ACCURATELY MEASURE AND MONITOR THE RELIABILITY OF SOFTWARE COMPONENTS. PREDICTED ON THE CONCEPT THAT EXECUTION TIME IS THE BEST PRACTICAL MEASURE FOR CHARACTERIZING THE STRESS PLACED ON SOFTWARE, PROVIDED THAT THE EXECUTION ENVIRONMENT IS REPRESENTATIVE. (DAN 244)

EXECUTIVE PROGRAM
A COMPUTER PROGRAM WHICH INVOKES AND CONTROLS BOTH HARDWARE AND SOFTWARE ELEMENTS ATTENDANT TO THE EXECUTION OF AN APPLICATIONS PROGRAM. (NASA) (2) A CENTRALIZED SUBPROGRAM WHICH PROVIDES SYSTEM USE FUNCTIONS SUCH AS CONTROL OF INPUT/OUTPUT AND SCHEDULING OF USE OF THE PROCESSOR(S) AND ALLOCATION OF COMPUTER RESOURCES INCLUDING PROCESSING TIME, MEMORY ACCESS, AND INTERRUPT PROCESSOR SERVICES REQUIRED BY APPLICATION SUBPROGRAMS AND THEIR PROGRAMMED TASKS. (DAN 1201)

EXIT
AN EXIT IS THE PLACE WHERE CONTROL LEAVES A ROUTINE. (SET)

EXPONENTIAL DISTRIBUTION
INDEXING TERM. REFERS TO THE MATHEMATICAL METHODOLOGY WHICH IS USED TO CONSTRUCT, OR WHAT IS THE FORM ASSUMED BY, A PARTICULAR MODEL.

EXPOSURE
THE DEGREE OF PROTECTION WHICH HAS BEEN PROVIDED FOR AN INDIVIDUAL OBJECT.

(DAN 277)

EXPRESSION OF A PROGRAM
A STATEMENT OF A PROGRAM IN A LANGUAGE. THE EXPRESSION OF A PROGRAM IN A PROGRAMMING LANGUAGE IS AN ORGANIZED COLLECTION OF SYMBOLS WHICH, ACCORDING TO THE LOGICAL, SYNTACTIC AND SEMANTIC RULES OF THE PROGRAMMING LANGUAGE, CHARACTERIZE THE ALGORITHM AND DATA OBJECTS WHICH MAKE UP THE PROGRAM. (ABBOTT)

EXTENSIBILITY
THE EXTENT TO WHICH SOFTWARE ALLOWS NEW CAPABILITIES TO BE ADDED AND EXISTING CAPABILITIES TO BE EASILY TAILORED TO USER NEEDS.

EXTERNAL ENVIRONMENT
THE COMBINATION OF HARDWARE AND SOFTWARE USED TO MAINTAIN AND EXECUTE THE SOFTWARE, INCLUDING THE COMPUTER ON WHICH THE SOFTWARE EXECUTES, THE OPERATING SYSTEM FOR THAT COMPUTER, SUPPORT LIBRARIES, TEXT EDITORS, COMPILER, ETC. (SEL)

FAILURE
A SOFTWARE FAILURE IS AN UNACCEPTABLE RESULT PRODUCED DURING THE OPERATION OF THE COMPUTER PROGRAM. A FAILURE OCCURS WHEN A FAULT IS EVOKED BY SOME INPUT DATA.

FAILURE CATEGORIZATION
ASSIGNING A FAILURE TO A DESCRIPTIVE CATEGORY BASED UPON THE TIME IN THE LIFE CYCLE THE FAILURE OCCURRED, THE MANIFESTATION OF THE FAILURE, AND THE CAUSE OF THE FAILURE. BY "CAUSE" IS MEANT THE ERROR WHICH WAS AT THE ROOT OF THE FAILURE.

FAILURE DATA
INPUTS TO A RELIABILITY ESTIMATION PROGRAM WHICH USUALLY INCLUDE A SET OF EXECUTION TIME INTERVALS BETWEEN FAILURES ALONG WITH THE NUMBER OF DAYS FROM THE START OF TESTING ON WHICH THE FAILURES OCCURRED, MEAN TIME BETWEEN FAILURES, EXECUTION TIME BETWEEN FAILURES.

FAILURE RATE
NUMBER OF FAILURES DIVIDED BY CPU TIME FOR THE INTERVAL. (DAN 226)

FAILURE RATIO
NUMBER OF FAILURES PER CALENDAR INTERVALS DIVIDED BY TOTAL NUMBER OF RUNS. (DAN 226)

FAIRNESS
A QUEUING SYSTEM IS CALLED FAIR IF, WHENEVER A PROCESS IS DELAYED ON ANY QUEUE, THERE IS A POSSIBLE FUTURE STATE OF THE SYSTEM IN WHICH THAT PROCESS IS ON NONE OF THE QUEUES. (DAN 420)

FAST (FORTRAN ANALYSIS SYSTEM)
A SECOND GENERATION PROGRAM ANALYZER DESIGNED TO SUPPORT PROGRAM DEVELOPMENT, DEBUGGING AND MAINTENANCE. ITS THREE MAJOR ELEMENTS ARE: 1) A SCANNER/PARSER TO CONVERT PROGRAM TEXT INTO A PROGRAM DATABASE, 2) A DATA BASE SYSTEM WITH DATA STRUCTURES AND RETRIEVAL CAPABILITIES TO SUPPORT THE PROGRAM ANALYSIS QUERY SET, AND 3) A COMMAND/QUERY LANGUAGE INTERPRETER TO

SATISFY QUERIES AND TO GENERATE ANALYSES. (DAN 261)

FAULT
A FAULT IS A SPECIFIC MANIFESTATION OF AN ERROR. THE FAULT IS EVIDENCED WHEN ENTRY OF SOME INPUT DATA RESULTS IN THE PROGRAM FAILING TO CORRECTLY PERFORM THE REQUIRED FUNCTION. AN ERROR MAY BE THE CAUSE OF SEVERAL FAULTS. (2) A FAULT IS A MANIFESTATION OF AN ERROR IN PROGRAM CODE... THE FAULT IS EVIDENCED WHEN ENTRY OF SOME INPUT DATA RESULTS IN THE PROGRAM FAILING TO CORRECTLY PERFORM THE REQUIRED FUNCTION. FAULT AND "BUG" ARE SYNONYMOUS. (SET)

FAULT AVOIDANCE
SET OF PROCEDURES LEADING TO ATTAINMENT OF RELIABLE SYSTEMS: INCLUDES ACQUISTION OF MOST RELIABLE COMPONENTS WITHIN THE GIVEN COST AND PERFORMANCE CONSTRAINTS, USE OF THOROUGHLY REFINED TECHNIQUES FOR THE INTERCONNECTION OF COMPONENTS AND ASSEMBLY OF SUBSYSTEMS, PACKAGING OF HARDWARE TO SCREEN OUT EXPECTED FORMS OF INTERFERENCE, AND CARRYING OUT OF COMPREHENSIVE TESTING TO ELIMINATE HARDWARE AND SOFTWARE DESIGN FAULTS. (DAN 236)

FAULT TOLERANCE
USE OF PROTECTIVE REDUNDANCY. A SYSTEM CAN BE DESIGNED TO BE FAULT-TOLERANT BY INCORPORATING ADDITIONAL COMPONENTS AND ABNORMAL ALGORITHMS WHICH ATTEMPT TO INSURE THAT OCCURRENCES OF ERRONEOUS STATES DO NOT RESULT IN LATER SYSTEM FAILURES-A QUANTITATIVE PREDICTION OF SYSTEM RELIABILITY. (DAN 236)

FAULT-TOLERANT SOFTWARE
A SOFTWARE STRUCTURE EMPLOYING FUNCTIONALLY REDUNDANT ROUTINES WITH CONCURRENT ERROR DETECTION, AND PROVISIONS TO SWITCH FROM ONE ROUTINE TO A FUNCTIONAL ALTERNATE IN THE EVENT OF A DETECTED FAULT. (DAN 225)

FIDELITY
FIDELITY IS DEFINED AS THE ACCURACY WITH WHICH A GIVEN ALGORITHM IS MECHANIZED FOR A GIVEN OPERATING SYSTEM AND HARDWARE SYSTEM. (DAN 781)

FILE
A COLLECTION OF DATA TREATED AS A UNIT. (ANSI-X3H1)

FILE MANAGEMENT
THE ACTION OF PROVIDING AND CONTROLLING ACCESS TO FILES, DIRECTING THEIR MAINTENANCE AND CONTROLLING THE RESOURCES USED. (ANSI-X3H1)

FILE MODIFICATION
CHANGING THE INFORMATION IN A FILE, I.E. DELETING, UPDATING, EXTENDING.

FILE MODIFICATION PROTECTION
CONSTRAINT ON A FILE SYSTEM WHICH RESTRICTS THE FILE SYSTEM FROM MODIFYING THE FILE IN ANY WAY.

FIRMWARE
HARD-WIRED PROGRAMS WHICH INTERPRET MACHINE LANGUAGE INSTRUCTIONS AND DIRECT THE CORRESPONDING ELEMENTAL MACHINE OPERATIONS. (NASA) (2) AN EXECUTABLE DIGITAL PROGRAM WHICH IS FIXED IN THE MEMORY OF THE COMPUTING DEVICE WHICH WILL EXECUTE IT. (DAN 1201)

FLAG
A SIMPLE DATA STRUCTURE THAT DIRECTS THE FLOW OF CONTROL IN A PROGRAM. IF IT HAS A RANGE OF ONLY TWO VALUES, IT IS SOMETIMES CALLED A "BOOLEAN" OR "SWITCH". FLAGS USED SOLELY TO PERMIT A PROGRAM TO HAVE STRUCTURED CONTROL FLOW ARE CALLED STRUCTURE FLAGS. (DAN 1153)

FLEXIBILITY
THE TERM FLEXIBILITY IS USUALLY USED TO DENOTE THE EXISTENCE OF A RANGE OF CHOICES AVAILABLE TO A PROGRAMMER OR IMPLEMENTER - THE MORE CHOICES, THE GREATER THE FLEXIBILITY. FLEXIBILITY IS SOMETIMES REFERRED TO AS "GENERALITY" (DAN 470) (2) FLEXIBILITY IS USEFUL COMPLEXITY. (DAN 781)

FLIGHT CRITICAL
ESSENTIAL TO SAFETY OR FLYABILITY OF AN AIRPLANE. (NASA)

FLIGHT TEST
A TECHNIQUE USED TO DEMONSTRATE HARDWARE AND SOFTWARE PERFORMANCE IN ACTUAL SYSTEM OPERATION. (DAN 134)

FLIGHT-PHASE CRITICAL
ESSENTIAL TO SAFETY OR FLYABILITY OF AN AIRPLANE IN ONLY CERTAIN FLIGHT PHASES OR ENVIRONMENTS. (NASA)

FLOW CHART
A GRAPHICAL REPRESENTATION FOR THE DEFINITION, ANALYSIS, OR SOLUTION OF A PROBLEM, IN WHICH SYMBOLS ARE USED TO REPRESENT OPERATIONS, DATA, FLOW, EQUIPMENT, ETC. (ANSI-X3) (2) A DIACRAM OF A PROGRAM'S LOGIC FLOW. (DAN LD7)

FLOW OF CONTROL
FLOW OF CONTROL IS THE ORDERED SEQUENCE OF OPERATIONS PERFORMED IN THE EXECUTION OF A SERIES OF ALGORITHMS...THE CONTROL STRUCTURES OF A HIGH-LEVEL PROGRAMMING LANGUAGE (FORTRAN, COBOL, PL/1, ETC.) ALLOW SEQUENTIAL PROCESSING AND BRANCHING. EXAMPLES OF FLOW-OF-CONTROL STATEMENTS IN A HIGH-LEVEL PROGRAMMING LANGUAGE ARE: GOTO, CASE, WHILE, IF-THEN-ELSE, ETC....ALSO SEE - GOTO CONTROL STRUCTURES. (SET)

FLOWCHARTER
A COMPUTER PROGRAM USED TO SHOW IN DETAIL THE LOGICAL STRUCTURE OF A COMPUTER PROGRAM. THE FLOW IS DETERMINED FROM THE ACTUAL OPERATIONS AS SPECIFIED BY THE EXECUTABLE STATEMENTS, NOT FROM COMMENTS. THE FLOWCHARTS GENERATED CAN BE COMPARED TO FLOWCHARTS PROVIDED IN THE COMPUTER PROGRAM SPECIFICATION TO SHOW DISCREPANCIES AND ILLUMINATE DIFFERENCES. (DAN 134)

FLOWCHARTING TOOLS
UTILITY PROGRAMS WHICH AUTOMATICALLY DRAW PROGRAM FLOWCHARTS DIRECTLY FROM SOURCE CODE. (DAN 142)

FOREIGN DEBUG
FOREIGN DEBUGGING (FD) IS AN IN-DEPTH PROGRAM REVIEW CONDUCTED BY SOMEONE OTHER THAN THE IMPLEMENTOR TO FIND PROGRAM ERRORS AND IMPROVE PROGRAM RELIABILITY...A NON-IMPLEMENTOR LEARNS THE INTERNAL CHARACTERISTICS OF THE PROGRAM TO BE DEBUGGED, CONSTRUCTS APPROPRIATE TEST CASES, AND DEBUGS JUST AS THE IMPLEMENTOR WOULD..ALSO SEE - PEER CODE REVIEW. (SET)

FORMAL SPECIFICATIONS
SOME SPECIFICATION TECHNIQUE BASED UPON A STRICT SET OF RULES FOR DESCRIBING THE SPECIFICATION AND USUALLY INVOLVING THE USE OF AN UNAMBIGUOUSLY DEFINED NOTATION (E.G., MATHEMATICAL FUNCTIONS, FORMAL PDL, ETC.). (SEL)

FORMAL TESTING
TESTING CONDUCTED ACCORDING TO TEST PROCEDURES WHICH ARE DOCUMENTED AND APPROVED BY CONTRACTOR AND CUSTOMER. (DAN 21) (2) TESTING PERFORMED IN ACCORDANCE WITH CUSTOMER-APPROVED TEST PLANS. THIS TYPE OF TESTING VERIFIES THAT THE SOFTWARE SYSTEM IS OPERATING ACCORDING TO THE REQUIREMENTS OF THE DEVELOPMENT SPECIFICATIONS. FORMAL TESTING IS USUALLY PERFORMED DURING THE SYSTEM EVALUATION PHASE OF SOFTWARE DEVELOPMENT. TERMS USED IN THE LITERATURE TO DESCRIBE THIS TESTING INCLUDE: (1) SYSTEM INTEGRATION TESTING, (2) PROTOTYPE TESTING, (3) SYSTEM TESTING, (4) ACCEPTANCE TESTING. (DAN 154)

FORMAL VALIDATION
MATHEMATICAL TECHNIQUES FOR PROVING PROGRAM CORRECTNESS. (DAN LD4) (2) SYNONYM FOR CORRECTNESS PROOF.

FORTRAN
(FORMULA TRANSLATION) A PROGRAMMING LANGUAGE PRIMARILY USED TO EXPRESS COMPUTER PROGRAMS BY ARITHMETIC FORMULAS. (ANSI-X3) (2) A NON-BLOCK STRUCTURED HOL IN WIDESPREAD USE FOR TECHNOLOGICAL APPLICATIONS. (NASA)

FUNCTION
A MATHEMATICAL NOTATION USED TO SPECIFY THE SET OF INPUTS, THE SET OF OUTPUTS, AND THE RELATIONSHIP BETWEEN THE INPUTS AND OUTPUTS. (SEL) (2) A FUNCTION IS A SUBPROGRAM WHICH RETURNS A PARTICULAR VALUE THAT IS DEPENDENT UPON THE INDEPENDENT VALUE(S) GIVEN WITH THE CALLING INSTRUCTION. ..NORMALLY THE VALUE RETURNED BY A FUNCTION IS DIRECTLY ASSOCIATED WITH THE NAME OF THE FUNCTION SUCH AS SIN(K). (SET) (3) A GROUPING OF ROUTINES WHICH PERFORMS A PRESCRIBED FUNCTION. (DAN 21) (4) A SUB-DIVISION OF PROCESSES. (DAN LD7) (5) IN COMPUTER PROGRAMMING, SYNONYM FOR PROCEDURE. (6) A PURPOSEFUL ROLE OR ACTION BASED ON A SPECIFIED RELATIONSHIP BETWEEN CIRCUMSTANCES AND RESPONSES. (NASA) (7) THE NATURAL, REQUIRED, OR EXPECTED ACTIVITY OF A PROGRAM ELEMENT IN CARRYING OUT A PROGRAM REQUIREMENT. (DAN 1201)

FUNCTION TESTING
THE PURPOSE OF THE FUNCTION TEST IS TO FIND DISCREPANCIES BETWEEN THE PROGRAM AND ITS EXTERNAL SPECIFICATION. (DAN 286) SEE ALSO: FUNCTIONAL TESTING

FUNCTIONAL INTEGRATION
JUDICIOUS COMBINATION OF RELATED INFORMATION, PROCESSES, OR OPERATIONS INTO A SYSTEM WHICH REALIZES FUNCTIONAL OBJECTIVES MORE EFFECTIVELY. (NASA)

FUNCTIONAL PROGRAMMING
A PROGRAMMING METHOLOLOGY AND THEORY OF PROGRAMMING BASED UPON THE SEMANOL DEFINITION OF A PROGRAM, "A PROGRAM P SPECIFIES A COMPUTABLE FUNCTION F ON THE SET E OF INPUTS SPECIFIED BY THE INPUT EXPRESSIONS IN THE PROGRAM". FUNCTIONAL PROGRAMMING INVOLVES EXPLICIT DEFINITION OF THE FUNCTIONAL REQUIREMENTS OF THE PROGRAM AND PROVIDES A METHOD FOR DESIGNING THE PROGRAM SO THAT IT CONTAINS ONLY THE FUNCTIONAL CAPABILITIES CORRESPONDING TO THE FUNCTIONAL REQUIREMENTS AND NO OTHERS. (DAN 233)

FUNCTIONAL REQUIREMENTS
THAT PART OF THE REQUIREMENTS WHICH DESCRIBE THE FUNCTIONS THE SYSTEM MUST PERFORM. (DAN 141)

FUNCTIONAL REQUIREMENTS DOCUMENT (FRD)
A DOCUMENT STATING THE ESSENTIAL TECHNICAL FEATURES OF A NEEDED DATA PROCESSING CAPABILITY, ALONG WITH TECHNICAL CONSTRAINTS AND CONDITIONS TO BE MET, AND CRITERIA FOR ACCEPTABLE DELIVERY THAT CAN BE APPENDED TO OR MADE A PART OF THE SOFTWARE REQUIREMENTS DOCUMENT (SRD). (DAN 1153).

FUNCTIONAL SPECIFICATIONS
A SPECIFICATION OF A COMPONENT AS A SET OF FUNCTIONS DEFINING THE OUTPUT FOR ANY INPUT. THE SPECIFICATION EMPHASIZES WHAT THE PROGRAM IS TO DO, RATHER THAN HOW TO DO IT. HOWEVER, AN ALGORITHMIC SPECIFICATION CAN BY CONSIDERED FUNCTIONAL IF IT IS NOT USED TO DICTATE THE ACTUAL ALGORITHM TO BE USED. (SEL) (2) DESCRIBE A SYSTEM IN TERMS OF ITS PRINCIPAL FUNCTIONS AND THEIR INTERRELATIONSHIPS, I.E., THE FUNCTIONAL RELATIONSHIPS OF THE PARTS. (DAN LD-7)

FUNCTIONAL TESTING
THE EXECUTION OF INDEPENDENT TESTS DESIGNED TO DEMONSTRATE A SPECIFIC FUNCTIONAL CAPABILITY OF A PROGRAM OR A SOFTWARE SYSTEM. (DAN 154) (2) VALIDATION OF PROGRAM "FUNCTIONAL CORRECTNESS" BY EXECUTION UNDER CONTROLLED INPUT STIMULI. THIS TESTING ALSO GAUGES THE SENSITIVITY OF THE PROGRAM TO VARIATIONS OF THE INPUT PARAMETERS. (DAN 1153)

FUNCTIONALLY READY
THE CONDITION WHEREIN A SYSTEM, SUBSYSTEM, OR COMPONENT EXHIBITS NO FAULTS WHICH WOULD PRECLUDE THE INITIATION OR CONTINUANCE OF AN INTENDED OPERATION. (NASA)

GENERALITY
GENERALITY IS THE DEGREE TO WHICH A SYSTEM IS APPLICABLE IN DIFFERENT ENVIRONMENTS. (DAN 781) (2) COMPARE WITH PORTABILITY AND ADAPTABILITY.

GENERATOR
A GENERATOR PRODUCES TEST DATA OR TEST CASES TO EXERCISE THE TARGET SYSTEM. A GENERATOR IN THIS CASE IS DIFFERENTIATED FROM A SIMULATOR BECAUSE IT ACTUALLY CREATES TEST DATA USING NUMERICAL INTEGRATORS, RANDOM NUMBER GENERATORS, ETC. ONCE THE DATA ARE PRODUCED BY THE GENERATOR, A SIMULATOR MIGHT BE REQUIRED TO ROUTE THE DATA TO THE SYSTEM. GENERATORS ARE USEFUL IN A SYSTEM TEST ENVIRONMENT WHERE "LIVE" DATA IS NOT AVAILABLE. USEFUL OUTPUT OF A DATA GENERATOR ARE TAPES OF LOGGED DATA THAT CAN BE USED WITH A DATA REPLAY FACILITY FOR ESTABLISHING STANDARD TEST CASES. (DAN 134)

GOTO
IN A HIGH-LEVEL PROGRAMMING LANGUAGE (FORTRAN, COBOL, PL/1,ETC.) GOTO IS A STATEMENT WHICH TELLS THE COMPUTER WHERE THE SEQUENCE OF EXECUTION SHOULD CONTINUE...A GOTO STATEMENT NORMALLY TRANSFERS CONTROL OF THE SEQUENCE OF INSTRUCTIONS TO SOME OTHER POINT IN THE PROGRAM. THE GOTO STATEMENT BECAME A DEBATING POINT WHEN DIJKSTRA SAID IN 1965 THAT THE QUALITY OF A PROGRAMMER WAS INVERSELY PROPORTIONAL TO THE NUMBER OF GOTO STATEMENTS IN HIS PROGRAMS. OTHERS ARGUED FOR THE RETENTION OF THE GOTO STATEMENT BECAUSE OF ITS USEFULNESS IN A LIMITED NUMBER OF SITUATIONS...ALSO SEE - FLOW OF

CONTROL.(SET)

GYPSY
GYPSY IS BOTH A GENERAL PROGRAMMING LANGUAGE AND A SPECIFICATION LANGUAGE WHICH CAN BE USED FOR SYSTEMS PROGRAMMING WHICH REQUIRE CONCURRENT PROCESSING AND PROCESS SYNCHRONIZATION FACILITIES. BASED ON PASCAL. (DAN389)

HAL/S
A REAL-TIME HIGHER ORDER PROGRAMMING LANGUAGE ESPECIALLY SUITED FOR SPACE AND AIRCRAFT APPLICATIONS. HAL/S WAS DEVELOPED FOR NASA BY INTERMETRICS, INC. WITH THE OBJECTIVE OF IMPROVING THE RELIABILITY AND REDUCING THE COST OF PRODUCING AVIONICS SOFTWARE. (DAN 388)

HALSTEAD'S LAW
PARTIAL DEFINITION: A FORMULA FOR PROGRAM LENGTH BASED ON THE NUMBER OF DISTINCT OPERATOR TYPES AND THE NUMBER OF DISTINCT OPERAND TYPES. (DAN 299)

HARDEST FIRST
THE DESIGN (OR IMPLEMENTATION) OF THE MOST DIFFICULT ASPECTS OF THE SYSTEM FIRST. (SEL)

HARDEST-OUT PRINCIPLE
THE BUILDING OF A SYSTEM BEGINNING WITH THAT PART WHICH, IN THE FINAL ANALYSIS, WOULD HAVE PROVED TO POSSESS THE HIGHEST RISK TO PROGRAMMING IF NOT PERFORMED FIRST. AT EACH SUBSEQUENT STEP, THE NEXT A POSTERIORI MOST CRITICAL PART IS ADDED, UNTIL THE ENTIRE SOFTWARE PACKAGE IS COMPLETED. (DAN 1153)

HARDWARE
PHYSICAL AND ELECTRICAL COMPONENTS OF A COMPUTER SYSTEM THAT PERFORMS THE INTENT OF INSTRUCTIONS FETCHED FROM MEMORY.

HARDWARE MONITORS
A UNIT THAT OBTAINS SIGNALS FROM A HOST COMPUTER SYSTEM THROUGH PROBES ATTACHED DIRECTLY TO THE COMPUTER'S CIRCUITRY. THE SIGNALS OBTAINED ARE FED TO COUNTERS AND TIMERS AND ARE RECORDED. THESE DATA ARE REDUCED TO OBTAIN INFORMATION ABOUT CPU UTILIZATION, CHANNEL ACTIVITY, ETC. THESE DATA CAN BE USED TO IMPROVE BOTH SYSTEM AND PROGRAM PERFORMANCE. (DAN 134)

HARDWARE RELIABILITY
A MEASURE OF THE SUCCESS WITH WHICH THE HARDWARE IN A SYSTEM CONFORMS TO SOME AUTHORITATIVE SPECIFICATION OF ITS BEHAVIOR, A QUANTATIVE ASSESSMENT. (DAN 236)

HAZARD FUNCTION
THE PROBABILITY OF AN ERROR OCCURRING IN A GIVEN INFINITESIMAL TIME INTERVAL GIVEN THAT NO ERROR HAS OCCURRED PREVIOUSLY TO THAT INTERVAL. (DAN 235) (2) INSTANTANEOUS FAILURE RATE OF A SYSTEM.(MUSA'S MODEL) (DAN 238) (3) THE ERROR-RATE RELATIONSHIP (DAN 238)

HEURISTIC
AN EXPLORATORY METHOD OF PROBLEM SOLVING IN WHICH SOLUTIONS ARE DISCOVERED BY EVALUATION OF THE PROGRESS MADE TOWARD THE FINAL RESULT. CONTRAST WITH ALGORITHM. (DAN 1153)

HEURISTIC SEARCH
A TESTING METHOD WHICH ESTABLISHES A SET OF HEURISTICS OF THE FORM: (S)-->(A)-->(R). THE LEFT PART REPRESENTS A SET OF STRESS STATES (S) WHICH MUST BE TRUE FOR THE RULE TO BE USED. THE SET (S) REPRESENTS THE RELEVANCE OR JUSTIFICATION OF THE HEURISTIC AND CAN HAVE ANY NUMBER OF MEMBERS. THE MIDDLE PART (A) REPRESENTS A SET OF ACTIONS WHICH SHOULD BE EXECUTED TO DISTURB THE THREAT SCENARIO. THESE ACTIONS (A) ARE LIMITED BY CONSTRAINT CONDITIONS (C) DEFINED BY THE TEST ENGINEER,. (A NOT EQUAL TO 0). THE RIGHT PART REPRESENTS A SET OF RESULTS (R) WHICH SHOULD BE TRUE AFTER APPLICATION OF THE RULE AND EXECUTION OF THE SYSTEM. THE SET (R) IS USED TO EVALUATE THE HEURISTIC. (DAN 428)

HIERARCHIAL STRUCTURE
A HIERARCHICAL STRUCTURE IS AN ORGANIZATION OF SOFTWARE MODULES THAT CONSISTS OF A ROOT NODE...THIS TERM CAN BE APPLIED TO DATA AS WELL AS PROGRAM. THIS STRUCTURE IS ALSO KNOWN AS TREE STRUCTURE WITHOUT CYCLES. (SET)

HIERARCHY
A SERIES OF SUCCESSIVE TASKS OR ROUTINES IN A GRADED ORDER. (2) A STRUCTURE BY WHICH OBJECTS OR CLASSES OF OBJECTS ARE RANKED ACCORDING TO SOME SUBORDINATING PRINCIPLE OR SET OF PRINCIPLES. ONE COMMON REPRESENTATION OF A HIERARCHY IS THE DIRECTED TREE-GRAPH, IN WHICH THE ROOT NODE HEADS THE HIERARCHY, AND ALL OTHER OBJECTS ARE RANKED BY ORDER INTO LEVELS OF SUBORDINATION. IF A SINGLE SUBORDINATING RELATIONSHIP GOVERNS THE HIERARCHY, IS SAID TO BE UNORDERED; OTHERWISE IT IS ORDERED. (DAN 1153)

HIGHER-ORDER LANGUAGE
A FULL REPERTOIRE OF INSTRUCTIONS AND STATEMENTS, HAVING FORMAL SYNTAX AND LEXICAL RULES, USABLE IN COMPOSING MACHINE-INDEPENDENT SOURCE PROGRAMS. (NASA) (2) SEE ALSO: HIGH LEVEL LANGUAGE (NOTE: SYNONOYMS?)

HIGH-LEVEL LANGUAGE
ISO A PROGRAMMING LANGUAGE THAT DOES NOT REFLECT THE STRUCTURE OF ANY ONE GIVEN COMPUTER OR THAT OF ANY GIVEN CLASS OF COMPUTERS.

HIPO (HIERARCHY PLUS INPUT-PROCESS-OUTPUT)
A GRAPHICAL TECHNIQUE THAT DEFINES EACH COMPONENT BY ITS TRANSFORMATION ON ITS INPUT DATA SETS TO ITS OUTPUT DATASETS.(SEL) (2) THIS PART DOCUMENTATION, PART ANALYTICAL TECHNIQUE CONSISTS OF HIERARCHY CHARTS AND THE CORRESPONDING INPUT-PROCESS CHARTS. THE HIERARCHY CHART IS A SET OF BLOCKS, SIMILAR TO AN ORGANIZATION CHART, SHOWING EACH FUNCTION AND ITS DIVISION INTO SUBFUNCTIONS. FOR EACH FUNCTION OR SUBFUNCTION, AN INPUT-PROCESS-OUTPUT CHART , ROUGHLY SIMILAR TO THE BLOCK DIAGRAM IN LOGIC DESIGN, SHOWS THE INPUTS AND OUTPUTS AND THE PROCESSES JOINING THEM. IF THE HIPO CHARTS THEMSELVES ARE ARRANGED IN A HIERARCHY, THE TECHNIQUES CAN BE USED TO GRAPHICALLY DOCUMENT TOP-DOWN DESIGN OR STRUCTURED DESIGN. (DAN 227) (3) HIERARCHY PLUS INPUT/PROCESS/OUTPUT IS A GRAPHIC DESIGN TECHNIQUE USED TO SHOW FUNCTION. HIPO DIAGRAMS DESCRIBE FUNCTIONS IN TERMS OF THE INPUT TO A PROCESS. THEY SHOW A SYSTEM, SUBSYSTEM, OR PROGRAM FUNCTIONALLY, I.E. THE FUNCTIONS THAT IT PERFORMS, ANSWERING THE QUESTION "WHAT DOES IT DO". SINCE THESE DIAGRAMS ARE VISUAL, THEY ARE EASIER TO UNDERSTAND THAN MOST DOCUMENTATION WHICH IS NARRATIVE. ALTHOUGH FLOWCHARTS ARE ANOTHER GRAPHIC DESIGN TECHNIQUE, THEY SHOW ORGANIZATION AND LOGIC IN CONTRAST TO FUNCTION.

(DAN 813)

HOL
ACRONYM FOR HIGH(ER) ORDER (OR LEVEL) LANGUAGE.

HOLDET
A LANGUAGE EVALUATION TOOL DEVELOPED BY MCDONNELL DOUGLAS ASTRONAUTICS CO. FOR THE U.S. ARMY FRANKFORT ARSENAL. IT IS ESSENTIALLY COMPRISED OF TWO PROCESSORS: HOLDEF PROCESSOR (HOLDEF IS THE DEFINITION LANGUAGE) AND A GENERALIZED TRANSLATOR (OPTRAN) (DAN 390)

HOST MACHINE
A GENERALLY MORE POWERFUL COMPUTER WHICH HELPS GENERATE OR WHICH EXECUTES THE SOFTWARE FOR ANOTHER COMPUTER, VIZ., THE TARGET COMPUTER. (NASA) (2) IN ARPANET TERMINOLOGY, THE "HOST MACHINE" IS THE COMPUTER THE USER IS CONNECTED TO THROUGH A TIP OR IMP FROM A DISTANT TERMINAL.

HUMAN ENGINEERING
CODE POSSESSES THE CHARACTERISTIC HUMAN ENGINEERING TO THE EXTENT THAT IT FULFILLS ITS PURPOSE WITHOUT WASTING THE USERS' TIME AND ENERGY, OR DEGRADING THEIR MORALE. THIS CHARACTERISTIC IMPLIES ACCESSIBILITY, ROBUSTNESS, AND COMMUNICATIVENESS. (DAN 239)

HYPERGEOMETRIC DISTRIBUTION
RELIABILITY MODEL THAT CAN BE USED TO ESTIMATE THE NUMBER OF REMAINING ERRORS IN A PROGRAM BY DELIBERATELY SEEDING NEW ERRORS, AND THEN HAVE SOMEONE ELSE FIND THE SEEDED ERRORS AS WELL AS THE INDIGENOUS OR UNDETECTED ERRORS STILL IN THE PROGRAM. (DAN 238) SEE ALSO: BUG SEEDING/TAGGING

IDENTIFIER
A SYMBOL WHOSE PURPOSE IS TO IDENTIFY, INDICATE, NAME, OR LOCATE A DATA STRUCTURE OR PROCEDURE ENTRY POINT. (DAN 1153)

IMP
SEE: INTERFACE MESSAGE PROCESSOR

IMPERFECT DEBUGGING
AN ASSUMPTION THAT ERRORS ARE NOT ALWAYS REMOVED OR CORRECTED WHEN DETECTED. (DAN 296)

IMPLEMENTATION
THE IMPLEMENTATION OF A PROGRAM IS EITHER A MACHINE EXECUTABLE FORM OF THE PROGRAM, OR A FORM OF THE PROGRAM THAT CAN BE AUTOMATICALLY TRANSLATED (E.G., BY COMPILER OR ASSEMBLER) (SEL) (2) THAT PROCESS BY WHICH AN ARCHITECTURAL DESIGN IS TURNED INTO A DELIVERED PROGRAM. IT INCLUDES THE DETAILED FUNCTIONAL AND PROCEDURAL DESIGN, CODING, TESTING, AND DOCUMENTATION NECESSARY TO MEET PROGRAM REQUIREMENTS, EITHER FOR NEW OR MODIFIED SOFTWARE. (DAN 1153)

IMPLEMENTATION CORRECTNESS
CORRECTNESS BETWEEN DESIGN AND PROGRAMMED HARDWARE (DAN 322)

IMPLEMENTATION MISINTERPRETATION ERROR
AN ERROR FOR A UNIT OF SOURCE CODE ASSOCIATED WITH A PROGRAM DUE TO THE

MISINTERPRETATION OF THE PROGRAM SPECIFICATIONS. (DAN 137)

IMPLEMENTATION PHASE
THE IMPLEMENTATION PHASE CONSISTS OF THE ACTUAL PROGRAM CODE GENERATION, UNIT TESTING OF THE PROGRAMS AND DOCUMENTING OF THE SYSTEM. (SET)

IMPLEMENTATION TECHNOLOGY
THE BODY OF TECHNOLOGY USEFUL IN THE DEFINITION, DESIGN, PROGRAMMING, AND PRODUCTION OF SOFTWARE. (NASA)

IMPLIED SYSTEM PERFORMANCE
AN UNWRITTEN REQUIREMENT WHICH IS UNDERSTOOD BY THE MAJORITY OF THE PROJECT TEAM TO BE ESSENTIALLY EQUIVALENT TO A WRITTEN REQUIREMENT. (DAN 31)

INDEPENDENT TEST TEAM
A PROJECT GROUP NOT ASSOCIATED WITH THE SOFTWARE DESIGN/DEVELOPMENT SECTION WHICH IS RESPONSIBLE FOR TESTING SOFTWARE TO CHECK ITS COMPLIANCE TO SPECIFICATIONS.

INDIGENOUS ERROR
AN ERROR EXISTING IN A PROGRAM THAT HAS NOT BEEN INSERTED FOR CALIBRATION PURPOSES. (DAN 1153)

INDUCTIVE ASSERTION
AN INVARIANT PREDICATE APPEARING WITHIN A PROCEDURE ITERATION. USUALLY PLACED JUST FOLLOWING THE LOOP-COLLECTING NODE, THESE PREDICATES ARE USED AS AN AID TOWARD PROVING CORRECTNESS. (DAN 1153) (2) A MATHEMATICAL OR LOGICAL DESCRIPTION OF THE STATE OF A COMPUTATION AT EACH INSTANCE OF AN EXECUTION THROUGH IT. IT TAKES THE FORM P <ASSERTION> Q WHERE P AND Q ARE PROGRAM SEGMENTS. FOR Q TO BE TRUE, P ACTING ON THE ASSERTIONS (WHICH ARE ALWAYS ASSUMED TRUE) MUST RESULT IN Q FOR THE ENTIRE DOMAIN SPECIFIED BY THE ASSERTIONS.

INFERENCE RULES
ANNOTATION TECHNIQUES/RULES USED IN PROVING PROGRAM CORRECTNESS. THE ANTECEDENTS OF EACH RULE ARE USUALLY ANNOTATED PROGRAM SEGMENTS CONTAINING INVARIANTS OR CANDIDATE INVARIANTS AND THE CONSEQUENT IS EITHER AN INVARIANT OR A CANDIDATE. DERSHOWITZ AND MANNA (REF 263) DIFFERENTIATE 3 TYPES OF RULES; ASSIGNMENT RULES, CONTROL RULES, AND HEURISTIC RULES. ASSIGNMENT RULES YIELD GLOBAL INVARIANTS BASED ONLY UPON THE ASSIGNMENT STATEMENTS OF THE PROGRAM. CONTROL RULES YIELD LOCAL INVARIANTS BASED UPON THE CONTROL STRUCTURES OF THE PROGRAM. HEURISTIC RULES HAVE CANDIDATES AS THEIR CONSEQUENTS. THESE CANDIDATES ARE NOT GUARANTEED TO BE INVARIANTS. (DAN 263)

INFORMAL PROOF OF CORRECTNESS
THE VISUAL INSPECTION OF A SMALL, COMPREHENSIVE SET OF TEST CASES INDICATING THAT THE CODE OF A PROGRAM SEGMENT MATCHES ITS SPECIFICATION. VALIDATION OF THE PROGRAM SEGMENT IS BASED ON AXIOMS STATING THAT LOWER-LEVEL SEGMENTS MATCH THEIR SPECIFICATIONS. (DAN LD7)

INFORMAL TESTING
TESTING THAT UTILIZES INTERNAL TEST DOCUMENTATION CONTROL AND PROCEDURES. INFORMAL TESTING USUALLY IS DESIGNED TO BE DEVELOPMENT GROUP TESTING AND REQUIRES NO FORMAL CUSTOMER APPROVAL. INFORMAL TESTING USUALLY BEGINS WHEN

THE FIRST PROGRAM UNIT IS CODED AND CONTINUES THROUGHOUT THE SYSTEM IMPLEMENTATION PHASE OF SOFTWARE DEVELOPMENT. TERMS USED IN THE LITERATURE TO DESCRIBE THIS TESTING INCLUDE: 1. UNIT TESTING, 2. SUBSYSTEM TESTING, 3. INTEGRATION TESTING, 4. COMPONENT TESTING, 5. DEVELOPMENT TESTING. (DAN 154) (2) TO TEST USING INFORMAL DOCUMENTATION. USUALLY A PRELIMINARY FORM OF TESTING PERFORMED BEFORE A FORMAL CERTIFICATION TEST. (DAN 1201)

INFORMATION

CORRELATION OF DATA FOR THE PROCESS OF INFORMING. (DAN 137) (2) A REPRESENTATION OF KNOWLEDGE, INTELLIGENCE, OR OTHER MEANINGFUL DATA IN A FORM THAT CAN BE USED TO CAUSE OR MODIFY THE PURPOSEFUL ACTIONS OF HUMANS OR MACHINES, PERHAPS AS THE RESULT OF PROPER ORGANIZATION, ANALYSIS, AND PRESENTATION. (DAN 1153) CONTRAST WITH DATA

INFORMATION HIDING

THE PROCESS OF ISOLATING CHANGEABLE PARTS OF A PROGRAM IN MODULES AND DEVELOPING AN INTERFACE BETWEEN THE MODULE AND THE REST OF THE PROGRAM THAT REMAINS VALID FOR ALL VERSIONS. (DAN 275) (2) A SOFTWARE DESIGN AND CODING CRITERION. A SYSTEM CONFORMS TO THE CRITERION OF INFORMATION HIDING TO THE EXTENT THAT ATTRIBUTES OF DATA OBJECTS ARE MANIPULATED INDIRECTLY VIA FUNCTIONS NAMED FOR THOSE ATTRIBUTES. A SYSTEM FAILS TO CONFORM TO THE CRITERION OF INFORMATION HIDING TO THE EXTENT THAT ATTRIBUTES OF DATA OBJECTS ARE MANIPULATED DIRECTLY IN TERMS OF IMPLICIT KNOWLEDGE OF THE REPRESENTATION OF THOSE ATTRIBUTES. (SEE REPRESENTATION). EXAMPLE. A THREE ELEMENT, ONE DIMENSIONAL, REAL ARRAY MAY BE USED TO REPRESENT THE (X,Y,Z) COORDINATE POSITION OF AN OBJECT IN SPACE. TO THE EXTENT THAT SUCH AN ARRAY IS OPERATED ON THROUGH THE USE OF ARRAY INDICES, (P(1), P(2), P(3)), THE SYSTEM IS NOT IN CONFORMANCE TO THE CRITERION OF INFORMATION HIDING. TO THE EXTENT THAT SUCH AN ARRAY IS OPERATED ON IN TERMS OF SYMBOLIC FUNCTIONS NAMED AFTER THE ATTRIBUTES, (X(P), Y(P), Z(P) OR P.X, P.Y, P.Z) THE SYSTEM IS IN CONFORMANCE TO THE CRITERION OF INFORMATION HIDING. (ABBOTT)

INFORMATION SYSTEMS

A SYSTEM WHICH PROVIDES PROCESSING CAPABILITIES FOR INFORMATION AND/OR DATA AND ALSO FOR MANAGING THE INFORMATION AND/OR DATA. (DAN 503) (2) AN ASSEMBLAGE OF METHODS, TECHNIQUES, PROCEDURES, PROGRAMS, OR DEVICES THAT SENSE, CONVEY, STORE, PROCESS, RETRIEVE, OR DISSEMINATE INFORMATION, UNITED BY REGULATED INTERACTION, TO ACCOMPLISH AN ORGANIZED, PURPOSEFUL TASK. (DAN 1153)

INFORMATION SYSTEMS TECHNOLOGY

THE BODY OF KNOWLEDGE AND PHYSICAL PHENOMENA THAT CONSTITUTE AN APPLIED SCIENCE ORIENTED TOWARD THE INDUSTRIAL USAGE OF INFORMATION SYSTEMS. (DAN 1153)

INITIATION

THE ACT OF CREATING AN ENVIRONMENT FOR THE INVOCATION OF AN ENTITY. (ANSI-X3H1)

INPUT ASSERTION

AN INPUT ASSERTION IS AN ASSERTION (USUALLY DENOTED BY THE GREEK LETTER PHI) THAT IMPOSES CONDITIONS ON THE INPUT TO A PROGRAM. IT IS USED TO SPECIFY THE DOMAIN OF INPUT VALUES OVER WHICH A PROGRAM IS INTENDED TO OPERATE. A PROGRAM IS SAID TO BE TOTALLY CORRECT WITH RESPECT TO AN INPUT ASSERTION

PHI, IF IT YIELDS THE DESIRED OUTPUT FOR ALL SETS OF INPUT VALUES SATISFYING PHI. ALSO SEE - ASSERTION, OUTPUT ASSERTION, PARTIAL CORRECTNESS, TOTAL CORRECTNESS. (SET)

INPUT - DATA SENSITIVITY
DEGREE TO WHICH PERFORMANCE IMPROVEMENTS DICTATED BY A PROGRAM MODIFICATION UNDER A CERTAIN SET OF INPUT DATA ARE PRESERVED UNDER DIFFERENT SETS OF INPUT DATA. (DAN 435)

INPUT/OUTPUT
(1) (ISO) PERTAINING TO A DEVICE OR TO A CHANNEL THAT MAY BE INVOLVED IN AN INPUT PROCESS AND, AT A DIFFERENT TIME, IN AN OUTPUT PROCESS. IN THE ENGLISH LANGUAGE, INPUT-OUTPUT MAY BE USED IN PLACE OF INPUT-OUTPUT DATA, INPUT-OUTPUT SIGNAL, INPUT-OUTPUT TERMINALS, ETC., WHEN SUCH USAGE IS CLEAR IN A GIVEN CONTEXT. (2) (ISO) PERTAINING TO A DEVICE WHOSE PARTS CAN BE PERFORMING AN INPUT PROCESS AND AN OUTPUT PROCESS AT THE SAME TIME. (3) PERTAINING TO EITHER INPUT OR OUTPUT, OR BOTH.

INSTALLATION DEFAULT
A DEFAULTED VALUE SPECIFIC TO A PARTICULAR SET OF HARDWARE, SOFTWARE AND PEOPLE. (ANSI-X3H1)

INSTRUCTION
AN ABSTRACT OR CONCRETE ENTITY THAT CAUSES A CHANGE IN STATE OR ACTION BY THE COMPUTER

INSTRUCTION SET
THE REPERTOIRE OF MACHINE LEVEL INSTRUCTIONS AVAILABLE TO A PROGRAMMER FOR A PARTICULAR COMPUTER. (NASA)

INSTRUCTION SIMULATOR
A COMPUTER PROGRAM USED TO SIMULATE THE EXECUTION CHARACTERISTICS OF A TARGET COMPUTER USING A SEQUENCE OF INSTRUCTIONS OF A HOST COMPUTER. THE INSTRUCTION SIMULATOR PROVIDES BIT-FOR-BIT FIDELITY WITH THE RESULTS THAT WOULD BE PRODUCED BY THE TARGET COMPUTER FOLLOWING THE SAME OPERATIONS AND INITIAL CONDITIONS. (DAN 134)

INSTRUCTION TRACE
A COMPUTER PROGRAM USED TO RECORD EVERY TIME A CERTAIN CLASS OF OPERATIONS OCCURS AND TRIGGER EVENT-DRIVEN DATA COLLECTION. IN SOME CASES, THIS CREATES A COMPLETE TIMED RECORD OF LITERALLY EVERYTHING SIGNIFICANT THAT OCCURRED DURING PROGRAM EXECUTION. THESE TRACES CONTAIN DATA ON INSTRUCTION AND BECOME A PERMANENT RECORD OF A PROGRAM'S EXECUTION. (DAN 134)

INSTRUMENTATION TOOLS
THOSE PROGRAMS THAT MONITOR AND RECORD INFORMATION ABOUT AN OBJECT SYSTEM OR PORTIONS THEREOF, AS IT OPERATES DATA REDUCTION AND ANALYSIS. (DAN LD7)

INTEGRATION
THE COMBINATION OF SUBUNITS INTO AN OVERALL UNIT OR SYSTEM BY MEANS OF INTERFACING IN ORDER TO PROVIDE AN ENVISIONED DATA PROCESSING CAPABILITY. (DAN 1153)

INTEGRATION TEST

INTEGRATION TEST - TEST OF SEVERAL MODULES IN ORDER TO CHECK THAT THE INTERFACES ARE DEFINED CORRECTLY. FULL INTEGRATION TEST - TESTING OF THE ENTIRE SYSTEM (I.E. TOP LEVEL COMPONENT). PARTIAL INTEGRATION TEST - TEST OF ANY SET OF MODULES PUT NOT THE ENTIRE SYSTEM. (SEL)

INTEGRITY PROBABILITY
A MEASURE OF SYSTEM SURVIVAL PROBABILITY. INTEGRITY PROBABILITY IS A FUNCTION OF SECURITY PROBABILITY, AND ATTACK PROBABILITY. SYSTEM SURVIVAL IS DEPENDENT ON THE FREQUENCY OF SYSTEM ATTACK COUPLED WITH THE ABILITY OF THE SYSTEM TO MAKE ITSELF SECURE FROM A PARTICULAR TYPE OF ATTACK. (DAN 781)

INTENDED USE OF
THE RESULT OF INVOKING A PROGRAM OR SEGMENT OF A PROGRAM, INCLUDING THE ACTIONS PERFORMED BY THAT PROGRAM WHEN INVOKED. INVOCATION MAY BE BY SUBROUTINE OR FUNCTION CALL, OR BY A BRANCH TO A SEGMENT OF CODE. (SEL)

INTERACTION
MUTUAL OR RECIPROCAL ACTION OR INFLUENCE BETWEEN TWO OR MORE ENTITIES. (ANSI-X3H1) (2) THE RECIPROCAL EFFECTS OF ONE SOFTWARE MODULE OR HARDWARE DEVICE ON ANOTHER. (DAN 1201)

INTERACTIVE
USAGE OF A COMPUTER VIA A TERMINAL WHERE EACH LINE OF INPUT IS IMMEDIATELY PROCESSED BY THE COMPUTER. (SEL)

INTERACTIVE DEBUG
INTERACTIVE DEBUGGING (ID) IS THE PROCESS OF SEEKING AND CORRECTING ERRORS IN A COMPUTER PROGRAM WHILE COMMUNICATING WITH THE COMPUTER EXECUTING THE PROGRAM...TYPICALLY, THE COMMUNICATION TAKES THE FORM OF MONITORING PROGRAM PROGRESS, INSPECTING INTERMEDIATE VALUES, INSERTING DATA CORRECTIONS AS NEEDED, AND, IN GENERAL, CONTROLLING PROGRAM EXECUTION. ID CAN DRAMATICALLY REDUCE THE TIME NEEDED TO DEBUG A PROGRAM SINCE THE PROGRAMMER CAN ACCOMPLISH IN A SHORT SESSION WITH THE "COMPUTER" (OFTEN, A REMOTE TERMINAL ATTACHED TO THE COMPUTER) WHAT WOULD NORMALLY TAKE SEVERAL BATCH TURNAROUNDS (E.G., IN MANY INSTALLATIONS, SEVERAL DAYS.) (SET)

INTERFACE
A SHARED BOUNDARY. AN INTERFACE MIGHT BE A HARDWARE COMPONENT TO LINK TWO DEVICES OR IT MIGHT BE A PORTION OF STORAGE OR REGISTERS ACCESSED BY TWO OR MORE COMPUTER PROGRAMS. (ANSI-X3) (2) INTERFACE - THE SET OF DATA PASSED BETWEEN TWO OR MORE PROGRAMS OR SEGMENTS OF PROGRAMS, AND THE ASSUMPTIONS MADE BY EACH PROGRAM ABOUT HOW THE OTHER(S) OPERATE. (SEL) (3) THE COMMON BOUNDARY BETWEEN SOFTWARE MODULES, BETWEEN HARDWARE DEVICES, OR BETWEEN HARDWARE AND SOFTWARE. (DAN 1201) (4) WHEN APPLIED TO A MODULE, THAT SET OF ASSUMPTIONS MADE CONCERNING THE MODULE BY THE REMAINING PROGRAM OR SYSTEM IN WHICH IT APPEARS. MODULES HAVE CONTROL, DATA, AND SERVICES INTERFACES. (DAN 1153)

INTERFACE CHECKER
A COMPUTER PROGRAM USED TO AUTOMATICALLY CHECK THE RANGE AND LIMITS OF VARIABLES AS WELL AS THE SCALING OF SOURCE PROGRAMS TO ASSURE FORMAT COMPLIANCE WITH INTERFACE AND CONTROL DOCUMENTS. (DAN 134)

INTERFACE CONTROL

INTERFACE CONTROL REQUIRES THAT INPUT/OUTPUT SPECIFICATIONS MUST BE CONTROLLED AS ENGINEERING CONFIGURATION ITEMS AT SYSTEM DESIGN, IMPLEMENTATION, INTEGRATION AND OPERATION TIMES. (DAN 346)

INTERFACE MESSAGE PROCESSOR (IMP)
A PIECE OF PACKET-SWITCHING HARDWARE USED FOR STORAGE OF MESSAGES, ROUTING OF SIGNALS AND PASSAGE OF COMMUNICATIONS IN THE ARPANET. IT IS SMALLER IN SCOPE IN ITS CAPABILITIES THAN THE TIP WHICH IT RESEMBLES FUNCTIONALLY.

INTERFACE SPECIFICATION DOCUMENT
A DOCUMENT THAT SERVES AS A COMMUNICATIONS VEHICLE BETWEEN THE CONFIGURATION CONTROL AND TECHNICAL IMPLEMENTATION PROCESSES, AND SUPPORTS THE COORDINATION OF EFFICIENT, CONTROLLABLE INTERFACES. (DAN LD7)

INTERFACE TESTING
VALIDATION THAT A MODULE OR SET OF MODULES OPERATE WITHIN AGREED INTERFACE SPECIFICATIONS TO ASSURE PROPER DATA AND LOGICAL COMMUNICATIONS. (DAN 1153)

INTERMITTENT ASSERTIONS METHOD
A TECHNIQUE OF PROVING TOTAL CORRECTNESS WHICH INVOLVES AFFIXING COMMENTS TO POINTS IN THE PROGRAM BUT WITH THE INTENTION THAT SOMETIME CONTROL WILL PASS THROUGH THE POINT AND SATISFY THE ATTACHED ASSERTION. CONSEQUENTLY, CONTROL MAY PASS THROUGH A POINT MANY TIMES WITHOUT SATISFYING THE ASSERTION, BUT CONTROL MUST PASS THROUGH THE POINT AT LEAST ONCE WITH THE ASSERTION SATISFIED; THEREFORE, WE CALL THESE COMMENTS INTERMITTENT ASSERTIONS. (DAN 419)

INTERNAL DELIVERY
THE POINT AT WHICH THE SOFTWARE AS AN ENTIRE PACKAGE IS GIVEN TO THE INDEPENDENT TEST GROUP. (DAN 21)

INTEROPERABILITY
THE TERM INTEROPERABILITY MEANS THAT ANY USER OF A LOCAL SYSTEM CAN POTENTIALLY ALSO OPERATE ANY INTERCONNECTED REMOTE SYSTEM. (DAN 346)

INTERPRET
EXECUTE MACHINE LANGUAGE PROGRAMS BY TRANSLATING EACH STATEMENT TO A CORRESPONDING SEQUENCE OF ELEMENTAL MACHINE OPERATIONS PRIOR TO PROCEEDING TO THE NEXT STATEMENT. (NASA)

INTERPRETATION CORRECTNESS
CORRECTNESS BETWEEN REQUIREMENTS AND SPECIFICATION (DAN 322)

INTERPRETIVE SIMULATOR
A HOST MACHINE PROGRAM WHICH INTERPRETIVELY EXECUTES A TARGET MACHINE PROGRAM IN A DYNAMICALLY REPRESENTATIVE MANNER, BUT WITHOUT ACKNOWLEDGING NUMERICAL OR ARITHMETIC EFFECTS. (NASA)

INTERPROCESS COMMUNICATION
THE SENDING AND RECEIVING OF MESSAGES BY THE PROCESSES/ENTITIES WITHIN AN OPERATING SYSTEM.

INTERRUPT
ANY STOPPING OF A PROCESS IN SUCH A WAY THAT IT CAN BE RESUMED. A PARTICULAR

TYPE OF INTERRUPT IS THE "TRAP". (DAN 1153)

INTERRUPT ANALYZER
A COMPUTER PROGRAM THAT EXAMINES SOURCE CODE AND DETERMINES POTENTIAL CONFLICTS IN THE USE OF DATA AND/OR STORAGE DUE TO INTERRUPTS. (DAN 134)

INVARIANT
AN INVARIANT IS AN ASSERTION ASSOCIATED WITH A POINT IN A PROGRAM THAT IS SATISFIED WHENEVER EXECUTION REACHES THAT POINT... AN INVARIANT THAT CUTS A LOOP IN THE PROGRAM IS SOMETIMES CALLED A "LOOP INVARIANT." SUCH AN ASSERTION IS SAID TO "CARRY" ITSELF AROUND THE LOOP...ALSO SEE - ASSERTION, LOOP ASSERTION. (SET)

INVARIANT ASSERTION
SYNONOMOUS WITH INVARIANT.

INVOCATION
THE TRANSFER OF CONTROL TO AN ENTITY CAUSING IT TO BE ACTIVATED. (ANSI-X3H1) (2) THE LINKING TO OR INSERTION OF A PROCEDURE BODY BY MEANS OF A NAMED REFERENCE WITHIN A PROCEDURE. SUBROUTINE LINKING IS SOMETIMES REFERRED TO AS A "CALL". CODE INSERTION IS REFERRED TO AS A "MACRO CALL". (DAN 1153)

INVOKE
THE ACT OF CAUSING INVOCATION.

I/O PROCESSOR
INPUT-OUTPUT PROCESSOR: A FRONT-END PIECE OF HARDWARE THAT INTERFACES BETWEEN THE INPUT-OUTPUT EQUIPMENT AND THE COMPUTER.

ISRAD
(INTEGRATED SOFTWARE RESEARCH AND DEVELOPMENT PROGRAM) A U.S. ARMY PROGRAM FOR RESEARCH AND DEVELOPMENT IN COMPUTER SCIENCE. (DAN 290)

ITERATION
AN ITERATION IS AN EXPRESSION IN A PROGRAMMING LANGUAGE THAT CAUSES A SEQUENCE OF INSTRUCTIONS TO BE REPEATED, UNTIL A SPECIFIED SET OF CONDITIONS IS EITHER MET OR NOT REACHED...AN ITERATION IS ONE OF THE FUNDAMENTAL CONTROL STRUCTURES IN PROGRAMMING . IT IS AN ALLOWED CONSTRUCT IN STRUCTURED PROGRAMMING... ALSO SEE - STRUCTURED PROGRAMMING (SET)

ITERATIVE ENHANCEMENT
THE DESIGN (OR IMPLEMENTATION) OF SUCCESSIVE VERSIONS, EACH PRODUCING A USABLE SUBSET OF THE FINAL PRODUCT UNTIL THE ENTIRE SYSTEM IS FULLY DEVELOPED. (SEL)

JAVS (JOVIAL AUTOMATED VERIFICATION SYSTEM)
AN AUTOMATED TOOL FOR TESTING JOVIAL PROGRAMS. THE SYSTEM WAS DEVELOPED UNDER CONTRACTS FROM ROME AIR DEVELOPMENT CENTER, GRIFFISS AFB, NY

JELINSKI-MORANDA MODEL
THIS MODEL WAS DEVELOPED BY DR. P. MORANDA AND MR. Z. JELINSKI OF MCDONNELL DOUGLAS ASTRONAUTICS CO. THE MODEL ASSUMES AN EXPONENTIAL FORM OF THE PROBABILITY DENSITY FUNCTION FOR THE DISTRIBUTION OF SOFTWARE ERRORS DETECTED AS A FUNCTION OF CALENDAR TIME. THE BASIC ASSUMPTIONS OF THIS MODEL

ARE: 1) THE AMOUNT OF DEBUGGING TIME BETWEEN ERROR OCCURRENCES HAS AN EXPONENTIAL DISTRIBUTION WITH AN ERROR OCCURRENCE RATE (OR HAZARD FUNCTION) PROPORTIONAL TO THE NUMBER OF REMAINING EPRORS. 2) EACH ERROR DISCOVERED IS IMMEDIATELY REMOVED, THUS DECREASING THE TOTAL NUMBER OF ERRORS BY ONE. 3) THE FAILURE RATE BETWEEN ERRORS IS CONSTANT. (DAN 402)

JOB
COMPUTER JOB CONSISTING OF ONE OR MORE STEPS SUCH AS COMPILATION, ASSEMBLY OR UTILITY RUNS. (DAN 137)

JOB CONTROL LANGUAGE (JCL)
SEE COMMAND LANGUAGE, OPERATING SYSTEM COMMAND AND RESPONSE LANGUAGE (OSCRL). (ANSI-X3H1)

JOINT LOGISTICS COMMANDERS
A SUBGROUP COMPOSED OF REPRESENTATIVES FROM THE ARMY MATERIAL COMMAND, THE NAVY MATERIAL COMMAND, AIR FORCE LOGISTICS COMMAND AND THE AIR SYSTEMS COMMAND. (DAN 222)

JOVIAL
CLASS NAME FOR A SET OF PROGRAMMING LANGUAGES ORIENTED TOWARD COMMAND AND CONTROL USAGE THAT ARE HIGHLY DISTINGUISHABLE BETWEEN EACH OTHER IN COMMANDS, SCOPE, AND FORMAT. THE LANGUAGES ARE MORE SPECIFICALLY TERMED BY THEIR SUFFIXES: JOVIAL(J3),J3; JOVIAL(J3B),J3B; JOVIAL(J73),J73.

KERNEL
A SMALL SELF-CONTAINED COLLECTION OF KEY SECURITY-RELATED STATEMENTS THAT WORKS AS A PRIVILEGED PART OF THE OPERATING SYSTEM AND ARE THEREBY ACCESSIBLE ONLY AT A HIGHER ASSESSIBILITY LEVEL THAN THE SUPERVISOR STATE. ALL CRITERIA SPECIFIED BY THE KERNEL MUST BE MET FOR A PROGRAM TO PERFORM SATISFACTORILY.

KNOT
A POINT AT WHICH TWO (OR MORE) DIRECTIONAL LINES, EACH INDICATING A FLOW OF CONTROL WITHIN A PROGRAM, ARE FORCED TO CROSS EACH OTHER. (USED TO DERIVE A PROGRAM COMPLEXITY MEASURE). (DAN 871)

LANGUAGE DESIGN
INDEXING TERM. REFERS TO THE PROCESS OF DESIGNING AND DEVELOPING A COMPUTER LANGUAGE OR TO AN ANALYSIS OF THE DESIGN OF A COMPUTER LANGUAGE.

LANGUAGE EVALUATION
INDEXING TERM. REFERS TO DOCUMENTS EVALUATING SPECIFIC FEATURES OF A COMPUTER LANGUAGE, OR COMPARING TWO OR MORE LANGUAGES IN GENERAL OR WITH RESPECT TO SPECIFIC FEATURES OR APPLICATIONS.

LANGUAGE PROCESSORS
COMPUTER PROGRAMS USED TO TRANSLATE HIGH-LEVEL OR SYMBOLIC INSTRUCTION MNEMONICS INTO COMPUTER-ORIENTED CODE CAPABLE OF BEING OBEYED BY A COMPUTER. THESE PROCESSORS TYPICALLY HAVE CAPABILITIES FOR ERROR DETECTION THROUGH SYNTAX ANALYSIS AND PROVIDE SYMBOLIC ADDRESSING, EXPRESSION EVALUATION, AND SYMBOL CROSS-REFERENCE LISTINGS. COMPILERS, ASSEMBLERS AND META-ASSEMBLERS ARE EXAMPLES OF THIS CATEGORY OF AIDS. (DAN 134)

LANGUAGE STRUCTURE
THE SET CONSISTING OF THOSE LANGUAGE CONSTRUCTS WHICH GOVERN FLOW OF CONTROL WITHIN A PROGRAM, MANAGEMENT OF MEMORY, CREATION OF DATA STRUCTURES AND EVALUATION OF EXPRESSIONS.

LEAST COMMON MECHANISM
THE RESULT OF APPLYING OCCAM'S RAZOR TO A SOLUTION. (ANSI-X3HI)

LEGIBILITY
CODE POSSESSES THE CHARACTERISTIC LEGIBILITY TO THE EXTENT THAT ITS FUNCTION IS EASILY DISCERNED BY READING THE CODE. (EXAMPLE: COMPLEX EXPRESSIONS HAVE MNEMONIC VARIABLE NAMES AND PARENTHESES EVEN IF UNNECESSARY.) LEGIBILITY IS NECESSARY FOR UNDERSTANDABILITY. (DAN 239

LEIGHTON DIAGRAM
A TOOL TO DISPLAY IN ONE PLACE THE RELATIONSHIP OF ALL PROCESSING ACTIVITIES, INPUTS, AND OUTPUTS,RELATING TO A SECTION OF A PROGRAM. (DAN 271)

LEVEL
A UNIT CORRESPONDING TO SOME PARTITIONING OF THE FINAL PRODUCT (E.G., A SINGLE LINE OF CODE, TEN LINES OF CODE, 25 LINES OF CODE, SUBROUTINE, MODULE). IF THE SYSTEM IS HIERARCHICALLY STRUCTURED, EACH COMPONENT IS AT A HIGHER LEVEL THAT ITS SUBCOMPONENTS, AND THE LEVEL MAY BE DESCRIBED AS THE HIGHEST LEVEL COMPONENT (THE COMPONENT AT LEVEL 1), OR THE COMPONENT AT LEVEL 2, OR THE LOWEST LEVEL COMPONENT, ETC. (SEL) (2) LOWEST LEVEL - SMALLEST UNIT IDENTIFIED BY THE ACTIVITY (E.G., CODE READING TO THE SINGLE STATEMENT, TOP DOWN DESIGN TO THE MODULE LEVEL, TOP DOWN DESIGN TO LEVEL 3). (SEL) (3) THE DEGREE OF SUBORDINATION IN A HIERARCHY. (DAN 1153)

LEVEL OF ACCESS
A SET OF FUNCTIONS, MACROS, SUBROUTINES, ETC., THAT ACCESS A PARTICULAR DATA STRUCTURE OR TYPE OF DATA STRUCTURE, THROUGH WHICH ALL ACCESSES TO THAT STRUCTURE OR TYPE, EXCEPT THOSE WITHIN THE FUNCTIONS, ETC., MUST PASS. ALSO CALLED "CLUSTERS" OR "PARNAS MODULES". (DAN 1153)

LEVEL OF NESTING
ONE MEASURE OF THE EXTENT TO WHICH NESTING IS USED IN AN OBJECT. THE LEVEL OF NESTING OF AN OBJECT IS DEFINED TO BE ONE GREATER THAN THE HIGHEST LEVEL OF NESTING USED IN ANY OF THAT OBJECT'S COMPONENT OBJECTS. IF AN OBJECT HAS NO COMPONENT OBJECTS, ITS LEVEL OF NESTING IS DEFINED TO BE ZERO.(ABBOTT)

LEVELS OF ABSTRACTION
A DESIGN AND IMPLEMENTATION METHOD THAT HELPS PRODUCE RELIABLE SOFTWARE THAT IS MORE EASILY MODIFIED AND MAINTAINED BY IDENTIFYING AND PLACING INTO A HIERARCHICAL STRUCTURE THE FUNCTIONAL PROCESSES AND DATA RESOURCES THAT CONSTITUTE THE SYSTEM'S PROGRAM STRUCTURE. IN ADDITION, A DESIGN DISCIPLINE THAT SUPPORTS THE CREATION OF A WELL BEHAVED, HIERARCHICALLY STRUCTURED, MODULAR SYSTEM. (DAN LD7) (2) LEVEL OF ABSTRACTIONS REFERS TO THE WAY IN WHICH A SPECIFIC SINGULAR COMPUTATION CAN BE INTELLECTUALLY GRASPED BY CONSIDERING IT AS A MEMBER OF LARGER CLASS OF DIFFERENT COMPUTATIONS...THIS TERM IS USUALLY USED DURING THE DESIGN AND CODING STAGE. THE TECHNIQUE TO ACHIEVE LEVELS OF ABSTRACTION IS TOP-DOWN DESIGN...ALSO SEE - TOP-DOWN, HIERARCHICAL STRUCTURE (SET) (3) A COLLECTION OF OPERATIONS, DATA OBJECTS

AND DATA TYPES USED TO DEFINE AN ABSTRACT MACHINE. (ABBOTT)

LEXICAL BINDING
LOCATION OF COMPONENTS CONSTITUTING A MODULE PHYSICALLY TOGETHER. (DAN 1153)

LIBRARIAN
A CLERK WHOSE RESPONSIBILITIES INCLUDE PROCESSING SOURCE STATEMENTS, BUT NOT WRITING THEM (E.G., MAINTAINING LIBRARIES, UPDATING CODE, PRODUCING TAPE BACKUPS, ETC.) (SEL) SEE ALSO DEVELOPMENT SUPPORT LIBRARIAN.

LIEN
A CHARGE UPON SOME DISCREPANT SOFTWARE ITEM IN THE FORM OF A DEBT OR DUTY LATER TO BE REDEEMED OR OTHERWISE SATISFIED. USUALLY THIS TERM REFERS TO THE DELIVERY OF SOFTWARE IN SOME USABLE FORM BUT REQUIRING THE REMOVAL OF DISCREPANCIES (PROGRAM OR DOCUMENTATION) IN ORDER TO BE COMPLETE. (DAN 1153)

LIKELIHOOD FUNCTION
SEE MAXIMUM LIKELIHOOD

LINE OF SOURCE CODE
80 CHARACTER CARD IMAGE OF SOURCE CODE. (DAN 137)

LINE OF SOURCE CODE FROM ANOTHER SOURCE
CODE NOT DEVELOPED BUT EXTRACTED FROM OTHER SOURCES. (DAN 137)

LINK
TO ESTABLISH CORRESPONDENCES WITHIN A SET OF CODE SEGMENTS WHICH SATISFY REFERENCES BETWEEN SEGMENTS. TO LINK-EDIT IS SYNONOMOUS WITH TO LINK. A LINKER OR LINK-EDITOR IS THE PROGRAM THAT CARRIES OUT THIS ACT. (ANSI-X3H1)

LINK EDITOR
A PROGRAM WHICH INTEGRATES SEPARATE RELOCATABLE CODE ROUTINES INTO A UNIFIED PROGRAM. (NASA) (2) SYNONYMOUS WITH LINKAGE EDITOR

LINKAGE EDITOR
A UTILITY ROUTINE THAT CREATES A LOADABLE COMPUTER PROGRAM BY COMBINING INDEPENDENTLY TRANSLATED COMPUTER PROGRAM MODULES AND BY RESOLVING CROSS REFERENCES AMONG THE MODULES. (ANSI-X3)

LISP LIST STRUCTURE
STRUCTURES FOR LISP LIST PROCESSING MECHANISM ARE COMPOSED OF LIST CELLS CONSISTING OF TWO POINTERS CALLED CAR AND CDR, WHICH MAY POINT TO OTHER LIST CELLS OR TO A VARIETY OF NON LIST OBJECTS. (DAN 872)

LIST PROCESSING
(ISO) A METHOD OF PROCESSING DATA IN THE FORM OF LISTS. USUALLY, CHAINED LISTS ARE USED SO THAT THE LOGICAL ORDER OF ITEMS CAN BE CHANGED WITHOUT ALTERING THEIR PHYSICAL LOCATIONS. (ANSI-X3)

LOADABLE PROGRAM DATA
DATA THAT IS RELOCATABLE OR ABSOLUTE BINARY MODULES PRODUCED BY A LINK EDITOR. (DAN LD7)

LOADER

A ROUTINE, COMMONLY A COMPUTER PROGRAM, THAT READS DATA INTO MAIN STORAGE. (2) A COMPUTER PROGRAM THAT ENABLES EXTERNAL REFERENCES OF SYMBOLS AMONG DIFFERENT ASSEMBLIES AS WELL AS THE ASSIGNMENT OF ABSOLUTE ADDRESSES TO RELOCATABLE STRINGS OF CODE. THIS PROGRAM PROVIDES DIAGNOSTICS ON ASSEMBLY OVERLAP, UNSATISFIED EXTERNAL REFERENCES, AND MULTIPLE DEFINED EXTERNAL SYMBOLS. (DAN 134) (3) A PROGRAM WHICH PRODUCES ABSOLUTE MACHINE CODE FROM A RELOCATABLE CODE OBJECT PROGRAM. (NASA)

LOGIC EQUATION GENERATOR
A COMPUTER PROGRAM USED TO AUTOMATICALLY RECONSTRUCT ARITHMETIC TEXT AND TO FLOWCHART ASSEMBLY LANGUAGE PROGRAMS. ONE SUCH PROGRAM TRANSLATES ASSEMBLY LANGUAGE INSTRUCTIONS INTO A MACHINE-INDEPENDENT MICROPROGRAMMING LANGUAGE AND BUILDS THE MICROPROGRAMMING STATEMENTS INTO A NETWORK IN WHICH FLOW OF CONTROL IS ANALYZED AND EQUATIONS RECONTRUCTED. (DAN 134)

LOGIC ERROR
AN ERROR IN A PROGRAM PROCEDURE, AS OPPOSED TO AN ERROR IN A PROGRAM FUNCTIONAL SPECIFICATION. (DAN 1153)

LOGICAL COMPLEXITY
LOGICAL COMPLEXITY IS A MEASURE OF THE DEGREE OF DECISION-MAKING LOGIC WITHIN A SYSTEM. (DAN 781) (2) PERHAPS SYNONOMOUS WITH COMPLEXITY. (3) THE DEGREE OF DECISION LOGIC IN A COMPUTER PROGRAM. (NASA)

LOGICWARE
THE LOGICAL SEQUENCE OF INSTRUCTIONS CONTROLLING THE EXECUTION SEQUENCE DONE BY THE HARDWARE. SEE ALSO DATAWARE. (DAN 781)

LOGISTICS APPLICATIONS
USE OF SOFTWARE SYSTEMS TO FACILITATE TRANSPORTATION AND SUPPLY AND THE MOVEMENT OF PERSONNEL IN ANY OF THE BRANCHES OF THE ARMED FORCES. (DAN 382)

LOOK-AHEAD DESIGN PRINCIPLE
THE PRINCIPAL BY WHICH A BASELINE OR PRELIMINARY DESIGN (OR PROGRAM ARCHITECTURE) IS DEVELOPED, WHICH IDENTIFIES AND SKETCHES THE KEY DETAILS OF THE REMAINING WORK TO BE DONE TO ASSURE THAT THE SUBSEQUENT DETAILED IMPLEMENTATION WILL BE PROPER WHEN VIEWED IN RETROSPECT. (DAN 1153)

LOOP
(ISO) A SET OF INSTRUCTIONS THAT MAY BE EXECUTED REPEATEDLY WHILE A CERTAIN CONDITION PREVAILS. IN SOME IMPLEMENTATIONS, NO TEST IS MADE TO DISCOVER WHETHER THE CONDITION PREVAILS UNTIL THE LOOP HAS BEEN EXECUTED ONCE.

LOOP ASSERTION
A LOOP ASSERTION IS AN ASSERTION ASSOCIATED WITH A POINT IN A PROGRAM LOOP. FLOYD'S METHOD REQUIRES THAT EVERY LOOP IN A PROGRAM TO BE VERIFIED BE "CUT" BY A LOOP ASSERTION...ALSO SEE - ASSERTION, INVARIANT (SET)

LOOP BODY
(1) THE PART OF A LOOP THAT ACCOMPLISHES ITS PRIMARY PURPOSE. (2) IN A COUNTER, A PART OF THE LOOP CONTROL. (3) CONTRAST WITH LOOP CONTROL.(ANSI-X3)

LOOP CONTROL

(1) THE PARTS OF A LOOP THAT MODIFY THE LOOP CONTROL VARIABLES AND DETERMINE WHETHER TO EXECUTE THE LOOP BODY OR EXIT FROM THE LOOP. (2) CONTRAST WITH LOOP BODY... SEE ALSO: CONTROL STRUCTURES. (ANSI-X3)

LOOP INITIALIZATION
THE PARTS OF A LOOP THAT SET ITS STARTING VALUES. (ANSI-X3)

LOOP-CONTROL VARIABLE
A VARIABLE THAT AFFECTS THE EXECUTION OF INSTRUCTIONS IN THE LOOP BODY AND IS MODIFIED BY A LOOP CONTROL. SEE ALSO: CONTROL STATEMENTS. (ANSI-X3)

MACHINE CYCLE
AN ALGORITHM WHICH DEFINES A SEQUENCE OF OPERATIONS PERFORMED BY AN ABSTRACT MACHINE TO DETERMINE WHICH ADDITIONAL OPERATIONS ARE TO BE PERFORMED AND TO WHICH DATA OBJECTS THEY ARE TO BE APPLIED. IF AN ABSTRACT MACHINE IS DEFINED AS HAVING A MACHINE CYCLE, THAT MACHINE CYCLE IS EXECUTED REGULARLY AND WHENEVER THE ABSTRACT MACHINE HAS COMPLETED THE PERFORMANCE OF THE OPERATION CALLED FOR BY THE PREVIOUS MACHINE CYCLE. (ABBOTT)

MACHINE LANGUAGE
A LANGUAGE, USING SEQUENCES OF 0'S AND 1'S TO CONVEY INFORMATION AND INSTRUCTIONS TO A COMPUTER, AND REQUIRING NO TRANSLATION PRIOR TO INTERPRETATION BY THE COMPUTER. (NASA)

MACHINE WORDS
NUMBER OF WORDS IN MAIN MEMORY THAT A COMPONENT OCCUPIES AT ONE TIME. (SEL)

MACRO
A MACRO IS A SINGLE INSTRUCTION IN A SOURCE LANAGUAGE THAT IS REPLACED BY A DEFINED SEQUENCE OF SOURCE INSTRUCTIONS IN THE SAME LANGUAGE. THE MACRO MAY ALSO SPECIFY VALUES FOR PARAMETERS IN THE INSTRUCTIONS THAT ARE TO REPLACE IT. DEFAULT VALUES MAY EXIST FOR THE PARAMETERS. (ANSI-X3H1) (2) A TEST REPLACEMENT MECHANISM WHEREBY A PREDEFINED SEQUENCE OF ASSEMBLY LANGUAGE STATEMENTS ARE INSERTED WHEREVER PRESCRIBED DURING THE TRANSLATION PROCESS. (NASA)

MACRO FACILITY
THE CAPABILITY TO DEFINE AND USE MACROS. (ANSI-X3H1)

MACROPROCESSORS
AN ASSEMBLY LANGUAGE MACROPROCESSOR ALLOWS A PROGRAMMER TO DEFINE A SEQUENCE OF STATEMENTS IN A PROGRAMMING LANGUAGE, CATALOG THE ENTIRE SEQUENCE UNDER A SINGLE NAME, AND LATER RETRIEVE THE SEQUENCE BY USING ONLY ITS NAME. THE SEQUENCE IS GENERALLY INSERTED DIRECTLY IN A PROGRAM AS PART OF THE ASSEMBLY PROCESS. FEATURES OF A MACROPROCESSOR MAY INCLUDE: RECOGNIZING THE OCCURRENCE OF MACRODEFINITIONS; DELETING MACRO DEFINITIONS FROM A TEXT STRING AND STORING THEM IN A TABLE; RECOGNIZING THE OCCURRENCE OF A MACRO-CALL SUBSTITUTING A MACRO BODY IN PLACE OF A NAME IN A TEXT STRING; HANDLING DUPLICATE DEFINITIONS; SIGNALLING AN ERROR FOR A MACRO CALL ON A NON-EXISTENT DEFINITION; PARAMETER PASSING; AND CONDITIONAL CAPABILITIES. (DAN 753)

MAIN
A MAIN IS A PROGRAM "UNIT" WHICHCONTAINS AT LEAST ON EXECUTABLE STATEMENT

AND WHICH HAS A STARTING ADDRESS FOR PROGRAM EXECUTION... NORMALLY THE "MAIN PROGRAM UNIT" IS THAT SET OF INSTRUCTIONS THAT DETERMINES THE BASIC SEQUENCE OF CONTROL. (SET)

MAINTAINABILITY

CODE POSSESSES THE CHARACTERISTIC MAINTAINABILITY TO THE EXTENT THAT IT FACILITATES UPDATING TO SATISFY NEW REQUIREMENTS OR TO CORRECT DEFICIENCIES. THIS IMPLIES THAT THE CODE IS UNDERSTANDABLE, TESTABLE AND MODIFIABLE; E.G. COMMENTS ARE USED TO LOCATE SUBROUTINE CALLS AND ENTRY POINTS VISUAL SEARCH FOR LOCATIONS OF BRANCHING STATEMENTS AND THEIR TARGETS IS FACILITATED BY SPECIAL FORMATS, OR THE PROGRAM IS DESIGNED TO FIT INTO AVAILABLE RESOURCES WITH PLENTY OF MARGINS TO AVOID MAJOR REDESIGN, ETC. (DAN 239) (2) MAINTAINABILITY IS THE PROBABILITY THAT, WHEN MAINTENANCE ACTION IS INITIATED UNDER STATED CONDITIONS, A FAILED SYSTEM WILL BE RESTORED TO OPERABLE CONDITION WITHIN A SPECIFIED TIME. DAN 781)

MAINTAINABILITY MEASUREMENT

THE PROBABILITY THAT WHEN MAINTENANCE ACTION IS INITIATED UNDER STATED CONDITIONS, A FAILED SYSTEM WILL BE RESTORED TO OPERABLE CONDITION WITHIN A SPECIFIED TIME. (DAN 233)

MAINTAINABLE

A SOFTWARE PRODUCT IS MAINTAINABLE TO THE EXTENT THAT IT CAN BE CHANGED TO SATISFY NEW REQUIREMENTS OR TO CORRECT DEFICIENCIES... SOME OF THE CHARACTERISTICS WHICH INDICATE THE EXTENT TO WHICH A SOFTWARE PRODUCT IS MAINTAINABLE ARE: (A) EASE OF MODIFYING ITS DOCUMENTATION; E.G. INSERTIONS AND DELETIONS CAN BE MADE WITHOUT RENUMBERING OTHER PAGES, AND REVISION RECORDS ARE AVAILABLE. (B) CODE MODIFICATIONS ARE TRACEABLE TO ANY PREVIOUS STATE (E.G. SOURCE CODE LINES SEQUENTIALLY NUMBERED, AND COMMENT MARKS USED TO CONVERT PREVIOUSLY EXECUTABLE SOURCE CODE STATEMENTS TO "COMMENTS" WHICH REMAIN IN THE LISTING AS A CHANGE RECORD). (C) DOCUMENTATION INCLUDES CROSS-REFERENCES OF VARIABLE NAMES WITH SUBROUTINES IN WHICH THEY ARE USED, AND SUBROUTINES CALLING SEQUENCES. (D) COMMENTS ARE USED TO LOCATE SUBROUTINE CALLS AND ENTRY POINTS. (E) SOURCE CODE FORMAT FACILITATES VISUAL SEARCH FOR LOCATIONS OF BRANCHING STATEMENT AND THEIR TARGETS. ALTERNATIVELY, UP-TO-DATE FLOWCHARTS ARE AVAILABLE. ALSO SEE - UNDERSTANDABLE, TESTABLE, AND MODIFIABLE. (SET)

MAINTENANCE

(ISO) ANY ACTIVITY, SUCH AS TESTS, MEASUREMENTS, REPLACEMENTS, ADJUSTMENTS, AND REPAIRS, INTENDED TO ELIMINATE FAULTS OR TO KEEP A FUNCTIONAL UNIT IN A SPECIFIED STATE. (ANSI-X3) (2) ACTIVITY WHICH INCLUDES THE DETECTION AND CORRECTION OF ERRORS AND THE INCORPORATION OF MODIFICATIONS TO ADD CAPABILITIES AND/OR IMPROVE PERFORMANCE. (SET) SEE ALSO PREVENTIVE MAINTENANCE, CORRECTIVE MAINTENANCE (3) SOFTWARE MAINTENANCE - THE PROCESS OF MODIFYING EXISTING OPERATIONAL SOFTWARE WHILE LEAVING ITS PRIMARY FUNCTION INTACT. (NASA) (4) ALTERATIONS TO SOFTWARE DURING THE POST-DELIVERY PERIOD IN THE FORM OF SUSTAINING ENGINEERING OR MODIFICATIONS NOT REQUIRING A REINITIATION OF THE SOFTWARE DEVELOPMENT CYCLE. (DAN 1153)

MAINTENANCE COSTS

FOR ERROR CORRECTION, PROGRAM MODIFICATION, OR ANY OTHER ACTIVITY REFERRED TO AS MAINTENANCE.

MAINTENANCE SOFTWARE
THE PORTIONS OF A DFCAS COMPUTER PROGRAM WHICH SUPPORT DFCAS MAINTENANCE, SUCH AS BY IDENTIFICATION AND ANNUNCIATION OF FAILED HARDWARE COMPONENTS. (NASA)

MAINTENANCE TOOLS
THE RESOURCES, (PERSONNEL TEST DRIVERS, SIMULATORS, ETC) NEEDED TO CARRY ON MAINTENANCE ACTIVITIES. (DAN 335)

MAJOR ERROR
A CATASTROPHIC EVENT WHICH INTERRUPTS OR COULD INTERRUPT MOST OR ALL MAJOR SYSTEM FUNCTIONS, E.G. AN INFINITE LOOP, SYSTEM CRASH, A MAJOR MEMORY OVERFLOW, A DATA BASE CORRUPTION, ETC. (DAN 31)

MANAGEMENT
SOFTWARE MANAGEMENT INCLUDES THE PHASES LIFE CYCLE ANALYSIS, REQUIREMENTS ANALYSIS, STRUCTURED DESIGN, EXTERNAL DOCUMENTATION, INTEGRATION OF MANAGERIAL AND TECHNICAL ISSUES. (DAN 273) (2) A TERM THAT INDICATES METHODOLOGY, TOOLS, AND PROCEDURES. (DAN LD7) (3) SOFTWARE MANAGEMENT CONSISTS OF ALL THE TECHNICAL AND MANAGEMENT ACTIVITIES, DECISIONS, AND CONTROLS THAT ARE DIRECTLY REQUIRED TO PURCHASE, PRODUCE, OR MAINTAIN SOFTWARE THROUGHOUT THE USEFUL LIFE OF A COMPUTER SYSTEM OR SERVICE. (DAN 1237)

MANAGEMENT CONTROL AND PROJECT VISIBILITY
THOSE PROCESSES THAT MONITOR THE PROJECT'S STATUS IN RESPECT TO PLANNED LEVELS OF SCHEDULE, COST, AND PERFORMANCE, AND TAKE CORRECTIVE ACTION IF NECESSARY. (DAN LD7)

MANAGEMENT FUNCTIONS
ALTHOUGH THE MANAGEMENT PROCESS HAS BEEN DESCRIBED IN MANY WAYS, FOUR BASIC FUNCTIONS HAVE RECEIVED GENERAL ACCEPTANCE - PLANNING, ORGANIZING, CONTROLLING AND COMMUNICATING. 1 PLANNING. THE PROCESS OF DETERMINING THE PROJECT OBJECTIVES AND THE POLICIES, PROGRAMS, PROCEDURES AND METHODS FOR ACHIEVING THEM. THE PLANNING FUNCTION MUST PROVIDE A FRAMEWORK FOR DECISION MAKING. 2. ORGANIZING. THE PROCESS OF DETERMINING THE ACTIVITIES REQUIRED TO ACHIEVE THE OBJECTIVES OF A PROGRAMMING PROJECT, THE DEPARTMENTATION OF THESE ACTIVITIES, AND THE ASSIGNMENT OF AUTHORITY AND RESPONSIBILITY FOR THEIR PERFORMANCE. 3. CONTROL. THE PROCESS OF ASSURING THAT THE VARIOUS COMPONENTS OF A PROJECT ARE PERFORMING IN ACCORDANCE WITH THE PLAN. CONTROL IS ESSENTIALLY THE MEASUREMENT AND MODIFICATION (IF NECESSARY) OF COMPONENT ACTIVITIES TO ASSURE THE ACCOMPLISHMENT OF THE OVERALL PLAN. 4. COMMUNICATIONS. THE PROCESS OF TRANSFERRING INFORMATION AMONG DECISION MAKERS THROUGHOUT THE PROJECT. (DAN141-MODIFIED)

MANAGEMENT STATISTICAL DATA
GENERAL NAME APPLIED TO ALL THE DATA COLLECTED AND ACCUMLATED BY THE PSL FOR THE PURPOSE OF PRODUCING MANAGEMENT REPORTS INCLUDING BOTH PLAN AND ACTUAL DATA. (DAN 137)

MANAGEMENT STATISTICAL DATA BASE
A DATA BASE CONTAINING MANAGEMENT STATISTICAL DATA FOR AN ONGOING PROGRAMMING PROJECT. (DAN 137)

MANAGEMENT TOOLS AND TECHNIQUES
ALL TOOLS AND TECHNIQUES UTILIZED IN CARRYING OUT THOSE MANAGEMENT FUNCTIONS REQUIRED TO OVERSEE THE DEVELOPMENT, MAINTENANCE, OR USE OF SOFTWARE.

MANPOWER
THE SUM, OVER THE NUMBER OF PEOPLE, OF THE NUMBER OF HOURS PER PERSON CHARGED TO THE CONTRACT, OR PROJECT. (SEL) (2) SEE ALSO MAN-UNITS.

MAN-DAY
SEE MAN-UNITS

MAN-HOUR
SEE MAN-UNITS

MAN-MONTH
SEE MAN-UNIT

MAN-UNITS
A CONCEPT USED TO ESTIMATE OR MEASURE HUMAN ENERGY TO BE EXPENDED OR WHICH HAS BEEN EXPENDED ON A PARTICULAR PROJECT. THE CONCEPT IS ULTIMATELY BASED ON THE LENGTH OF A WORKING DAY, 6 OR 8 HOURS (PRODUCTIVE TIME OR CALENDER TIME). THUS IF A MAN-DAY IS 6 HOURS, 5 DAYS OR 30 MAN-HOURS IS A MAN-WEEK; 48 MAN-WEEKS OR 1440 MAN-HOURS IS A MAN-YEAR; 4 MAN-WEEKS OR 120 MAN-HOURS IS A MAN-MONTH. A SIMILAR SET OF CORRESPONDENCES CAN BE CONSTRUCTED BASED ON 8 HOURS (OR ANY OTHER NUMBER) PER DAY.

MAN-YEAR
SEE MAN-UNIT

MANUAL-BASED TESTING
TESTING THAT IS USUALLY DIRECTED AT EVALUATING BOTH THE DESIGN AND THE PRODUCT (I.E. PROGRAMS AND DOCUMENTATION). THE DESIGN IS USUALLY EVALUATED FROM DOCUMENTS CONTAINING INFORMATION SUCH AS FUNCTIONAL REQUIREMENTS, SYSTEM SPECIFICATIONS, AND PROGRAM SPECIFICATIONS. THE PRODUCT EVALUATION USUALLY INVOLVES REVIEW OF THE COMPUTER PROGRAMS AND THE DOCUMENTATION DESCRIBING THE PROGRAMS OR SYSTEMS. (DAN 154)

MAP PROGRAM
A COMPUTER PROGRAM USED TO PROVIDE LOCATION AND/OR SIZE INFORMATION ABOUT ALL OR SELECTED PARTS OF THE TARGET SYSTEM, OR ABOUT DEVICE-RESIDENT DATA. (DAN 134)

MARKOV MODEL
INDEXING TERM. REFERS TO THE MATHEMATICAL METHODOLOGY WHICH IS USED TO CONSTRUCT, OR WHICH IS THE FORM ASSUMED BY, A PARTICULAR MODEL.

MATHEMATICAL/NUMERICAL
THIS CATEGORY OF SOFTWARE COMPONENTS IS MEANT TO BE A MORE SPECIFIC CATEGORY THAN THE SCIENTIFIC CLASS. IT CONTAINS THOSE COMPONENTS WHICH REFLECT A SPECIFIC ALGEBRAIC EXPRESSION OR MATHEMATICAL ALGORITHM. SUCH COMPONENTS AS A DOT PRODUCT ROUTINE OR A NUMERICAL INTEGRATOR FALL INTO THIS CATECORY. (SEL)

MAXIMUM LIKELIHOOD

A FUNCTION WHICH DESCRIBES THE NUMBER OF EXPECTED ERRORS LEFT IN A SOFTWARE PACKAGE TO A GIVEN LEVEL OF CONFIDENCE. THE FUNCTION IS BASED ON THE NUMBER OF ERRORS OBSERVED AND THE NUMBER CORRECTED. (DAN 296)

MAXIMUM LIKELIHOOD ESTIMATOR
THAT FUNCTION OF OBSERVED DATA THAT ESTIMATES AN UNKNOWN PARAMETER OF A KNOWN OR ASSUMED PROBABILITY DISTRIBUTION FUNCTION AS THE VALUE THAT MAXIMIZES THE PROBABILITY (DENSITY) FUNCTION ON THE OBSERVED DATA. (DAN 1153)

MAXIMUM SPACE
TOTAL AMOUNT OF MACHINE WORDS THAT THE SYSTEM MAY OCCUPY AT ONE TIME. (SEL)

MEASUREMENT
A NUMBER WITH AN ASSOCIATED UNIT OF MEASURE WHICH DESCRIBES SOME ASPECT OF SOFTWARE. SYNONOMOUS WITH METRIC.

MECHANICAL DEDUCTION
AUTOMATED OR SEMI-AUTOMATED VERIFICATION TOOL/TECHNIQUE WHICH IS USED TO PROVE PROGRAM CORRECTNESS.

MEMORY MANAGEMENT
A SYSTEM OF ADAPTIVE CONTROL THAT ALLOCATES MEMORY AND SCHEDULES THE CENTRAL PROCESSOR IN ORDER TO MAXIMIZE PERFORMANCE. (DAN 595)

MESSAGE
ANY COMMUNICATION SENT BETWEEN PERSONS OR PROCESSES. DATA INTENDED TO BE OR HAVING BEEN TRANSMITTED BETWEEN A SOURCE AND DESTINATION

MESSAGE SWITCHING
THE COMPUTER-CONTROLLED TRANSMISSION OF MESSAGES, BETWEEN TWO OR MORE POINTS, VIA COMMUNICATIONS FACILITIES, WHEREIN THE CONTENT OF THE MESSAGE REMAINS UNALTERED. (FCC)

MESSAGE TRANSFER MODEL
A MODEL WHICH DESCRIBES A COMPONENT OR MODULE IN TERMS OF ITS INTERACTIONS WITH OTHER COMPONENTS OR MODULES WHICH ARE AT ITS SAME LEVEL OF DECOMPOSITION. A MESSAGE TRANSFER MODEL CAN BE USED AS A DESIGN OR SPECIFICATION TECHNIQUE. (DAN 242)

METACOMPILERS
A COMPILER SYSTEM DESIGNED SPECIFICALLY TO IMPLEMENT (COMPILE) LANGUAGE COMPILERS. (DAN LD7)

METALANGUAGE
A METALANGUAGE IS A FORMAL MECHANISM USED TO DESCRIBE, SPECIFICALLY, OTHER LANGUAGES.

META-PROGRAMMING
THE PROCESS OF EXPRESSING PROBLEMS IN AN EXTENDED META-LANGUAGE, (LIKE BNF). (DAN 874)

METRIC
A MEASURE OF THE EXTENT OR DEGREE TO WHICH THE SOFTWARE POSSESSES AND

EXHIBITS A CERTAIN CHARACTERISTIC, QUALITY, PROPERTY, OR ATTRIBUTE. (2) A MEANINGFUL MEASURE OF THE EXTENT OR DEGREE TO WHICH AN ENTITY POSSESSES OR EXHIBITS A PARTICULAR CHARACTERISTIC. (NASA)

MICROCODE
A SET OF CONTROL FUNCTIONS PERFORMED BY THE INSTRUCTION DECODING AND EXECUTION LOGIC OF A COMPUTER WHICH DEFINES THE INSTRUCTION REPERTOIRE OF THAT COMPUTER. MICROCODE IS NOT GENERALLY ACCESSIBLE BY THE PROGRAMMER. (DAN 370)

MICROCOMPUTER
A CLASS OF COMPUTER HAVING ALL MAJOR CENTRAL PROCESSOR FUNCTIONS CONTAINED ON A SINGLE PRINTED CIRCIUT BOARD CONSTITUTING A STAND-ALONE MODULE. MICROCOMPUTERS ARE TYPICALLY IMPLEMENTED BY A SMALL NUMBER OF LSI CIRCIUTS AND ARE CHARACTERIZED BY A WORD SIZE NOT EXCEEDING 16 BITS, AND VERY LOW COST, USUALLY UNDER $1,000.

MICROPROCESSOR
A SINGLE LSI CIRCUIT WHICH PERFORMS THE FUNCTIONS OF A CPU. SOME CHARACTERISTICS OF A MICROPROCESSOR INCLUDE SMALL SIZE, INCLUSION IN A SINGLE INTEGRATED CIRCUIT OR A SET OF INTEGRATED CIRCUITS AND LOW COST. (DAN 370)

MICROPROGRAM
A PROGRAM IMPLEMENTED IN MICROCODE. (DAN 370) (2) A SEQUENCE OF INSTRUCTIONS, HARDWIRED IN A COMPUTER AND OPERATING ON INDIVIDUAL BITS OF DIGITAL WORDS, WHICH THE COMPUTER USES TO INTERPRET MACHINE LANGUAGE INSTRUCTIONS. (NASA)

MICRORELIABILITY MODEL
A RELIABILITY MODEL WHICH MEASURES THE RELIABILITY OF THE SEPARATE MODULES OF A PROGRAM BEFORE THE MODULES ARE COMBINED INTO A SOFTWARE SYSTEM. (DAN 299)

MINICOMPUTER
INDEXING TERM. MAY REFER TO DESIGN/DEVELOPMENT OF SOFTWARE FOR A MINICOMPUTER SYSTEM, OR TO THE USE OF MINICOMPUTERS IN A SOFTWARE DEVELOPMENT PROJECT (E.G., AS AN EMULATOR OR SIMULATOR), OR TO A COST-BENEFIT ANALYSIS OF MINICOMPUTERS VS. MAINFRAME COMPUTERS AS COMPONENTS OF A HARDWARE/SOFTWARE SYSTEM.

MINOR ERROR
A MARGINAL EVENT WHICH ALLOWS OR COULD ALLOW SOME PORTION OF THE SYSTEM TO OPERATE PROPERLY WHILE INTERRUPTING OTHERS, E.G. SOME MISSING OUTPUT, SOME WRONG OUTPUT, AN INACCURATE COMPUTATION, A RECOVERABLE TRANSIENT ERROR, ETC. (DAN 31)

MISSING PATH ERROR
AN ERROR IN WHICH A REQUIRED PREDICATE DOES NOT APPEAR IN THE GIVEN PROGRAM TO BE TESTED. ESPECIALLY IF THIS PREDICATE WERE AN EQUALITY, NO TESTING STRATEGY COULD SYSTEMATICALLY DETERMINE THAT SUCH A PREDICATE SHOULD BE PRESENT. (DAN 842)

MISSION DATE

DATE WHEN SYSTEM MUST BE OPERATIONAL. (SEL)

MISTAKE
A HUMAN ACTION PRODUCING AN UNINTENDED RESULT. (DAN LD4)

MODE
A WAY OF OPERATING A PROGRAM TO PERFORM A CERTAIN SUBSET OF THE FUNCTIONS THAT THE ENTIRE PROGRAM CAN PERFORM, AS SELECTED BY CONTROL DATA OR OPERATING CONDITIONS. OFTEN, THE MODE OF A PROGRAM WILL BE DEFINED AS PROGRAM STATES, WITH TRANSITIONS ANNOTATED TO DELINEATE EVENTS CAUSING THE PASSAGES BETWEEN MODES OF OPERATION. (DAN 1153)

MODEL
A MODEL IS AN ABSTRACTION OF A REAL WORLD PROCESS. (DAN 238) SEE ALSO: BEHAVIORAL MODEL, STRUCTURAL MODEL.

MODELING AND SIMULATION TOOLS
TOOLS USED FOR TRADE-OFF STUDIES AND TO INVESTIGATE PARTICULAR ABSTRACTIONS AND APPROCACHES FOR THE SYSTEM DESIGN. THEY ARE USEFUL FOR ANALYZING AND MODELING PARTICULAR APPROACHES TO SYSTEM DESIGNS. EXAMPLES INCLUDE: CASE, GPSS, MODLIT, SCERT, AND SPCL. (DAN LD7)

MODERN PROGRAMMING PRACTICES
A GENERAL TERM ENCOMPASSING VARIOUS PROCEDURES, STANDARDS, PROGRAMMING AND DESIGN TECHNIQUES WHICH EVOLVED THROUGH THE IMPETUS GENERATED BY THE MOVE TOWARD STRUCTURED PROGRAMMING. PROGRAMMING PRACTICES DESCRIBED AS "MODERN" USUALLY INCLUDE STRUCTURED PROGRAMMING, TOP-DOWN PROGRAM DESIGN, CHIEF PROGRAMMER TEAM, MODULAR PROGRAMMING, AND DEVELOPMENT SUPPORT LIBRARIAN. (2) A DYNAMICALLY CHANGING TERM WITH SELF-EXPLANATORY DEFINITION. AT PRESENT, 'MODERN PROGRAMMING PRACTICES' IS CONNOTATIVELY EQUIVALENT WITH USING HIERARCHIAL STRUCTURED-PROGRAMMING CONCEPTS, WITH, SOMETIMES, OCCASIONAL ASSOCIATION WITH EASILY VERIFIABLE CONSTRUCTS FROM THE LANGUAGE BEING USED.

MODIFIABILITY
CODE POSSESSES THE CHARACTERISTIC MODIFIABILITY TO THE EXTENT THAT IT FACILITATES THE INCORPORATION OF CHANGES, ONCE THE NATURE OF THE DESIRED CHANGE HAS BEEN DETERMINED. NOTE THE HIGHER LEVEL OF ABSTRACTNESS OF THIS CHARACTERISTIC AS COMPARED WITH AUGMENTABILITY. (2) MODIFIABILITY IMPLIES CONTROLLED CHANGE, IN WHICH SOME PARTS OR ASPECTS REMAIN THE SAME WHILE OTHERS ARE ALTERED, ALL IN SUCH A WAY THAT A DESIRED NEW RESULT IS OBTAINED. (DAN 109)

MODIFIABLE
MODIFIABILITY IS THE CHARACTERISTIC OF BEING EASY TO MODIFY...MODIFIABILITY OR TO BE MODIFIABLE IMPLIES CONTROLLED CHANGE IN WHICH SOME PARTS OR ASPECTS REMAIN THE SAME, WHILE OTHERS ARE ALTERED; ALL IN SUCH A WAY THAT A DESIRED NEW RESULT IS OBTAINED. MODIFIABILITY IS ONE ASPECT OF MAINTAINABLE. ALSO SEE - MAINTAINABLE. (SET)

MODIFICATION
THE PROCESS OF ALTERING A PROGRAM AND ITS SPECIFICATION SO AS TO PERFORM EITHER A NEW TASK OR A DIFFERENT BUT SIMILAR TASK. IN ALL CASES, THE FUNCTIONAL SCOPE OF A PROGRAM UNDER MODIFICATION CHANGES. (DAN 1153)

MODIFICATION PROCEDURES
THE FORMS, CHANNELS FOR APPROVAL, JUSTIFICATION, INITIATION, AND HANDLING OF REQUESTS FOR SOFTWARE MODIFICATION. (DAN 300)

MODULAR DECOMPOSITION
MODULAR DECOMPOSITION IS THE PROCESS OF BREAKING A LARGE PROGRAM INTO SMALL MODULES. ALSO SEE - MODULARITY, PROGRAM MODULE. (SET) (2) TO ISOLATE THE ENTIRE SYSTEM INTO INDEPENDENT PARTITIONS, EACH MODULE IS CONSTRUCTED TO WORK WITH OTHERS ON CONTROL SIGNALS AND DATA TRANSFERS, BUT TO BE UNINVOLVED IN THE DETAILED INTERNAL STRUCTURE OF OTHER MODULES. WITH INTER-MODULE INTERFACES CAREFULLY SPECIFIED, THE RELATIVELY INDEPENDENT MODULES BECOME EASIER TO CODE,, TEST, AND LATER CHANGE THAN MORE DEPENDENT MODULES. (DAN 227) (3) THE PROCESS OF BREAKING A LARGE PROGRAM INTO SMALL MODULES THAT PERFORM COMPLETE FUNCTIONS.

MODULAR PROGRAMMING
THE TECHNIQUE OF PRODUCING RELATIVELY SMALL, EASILY INTERCHANGEABLE COMPUTER ROUTINES WHICH MEET CERTAIN STANDARDIZED INTERFACE REQUIREMENTS. THIS TECHNIQUE MAKES IT EASIER TO DEVELOP AND VERIFY COMPLETED COMPUTER PROGRAMS. MODULARITY IS ACCOMPLISHED BY BREAKING THE PROGRAM INTO LIMITED LINE-SEGMENTS THAT PERFORM COMPLETE FUNCTIONS AND ARE THEREFORE, COMPLETELY UNDERSTANDABLE IN THEMSELVES. AIDS THAT HELP IMPLEMENT THESE TECHNIQUES ARE STANDARDS AND PROCEDURES. (DAN 134)

MODULARITY
MODULARITY IS THE FRAGMENTATION OF A PROGRAM INTO CONVENIENT DISCRETE PIECES CALLED MODULES...THE MAIN GOAL OF MODULARIZING A PROGRAM IS TO MAKE POSSIBLE THE MODIFICATION OF A SINGLE MODULE WITHOUT AFFECTING THE OTHER MODULES. IN THE CONTEXT OF SOFTWARE ENGINEERING, THIS IS CONSIDERED AS A QUALITY CHARACTERISTIC OF PROGRAMMING. THE CRUCIAL ELEMENTS ARE: A) SMALL (THE SIZE OF THE MODULE CANNOT BE QUANTIFIED AND MOST PROGRAMMERS PREFER TO FOLLOW THEIR OWN INTUITIVE APPROACH TO MODULARITY, B) SELF-CONTAINMENT, C) INDEPENDENCE (MEANING A PROGRAM IN WHICH ANY LOGICAL PORTION CAN BE CHANGED WITHOUT AFFECTING THE REST OF THE SYSTEM). ALSO A MODULAR PROGRAM SHOULD HAVE MODULES THAT HAVE ONLY ONE ENTRY POINT AND ONE EXIT POINT. ALSO SEE - PROGRAM MODULE (SET) (2) MODULARITY DEALS WITH HOW THE STRUCTURE OF AN OBJECT CAN MAKE THE ATTAINMENT OF SOME PURPOSE EASIER. MODULARITY IS PURPOSEFUL STRUCTURING. (DAN 109)

MODULARIZATION
REPRESENTING A SYSTEM AS A CONFIGURATION OF MODULES, WITH EACH MODULE BEING A LOGICAL CONFIGURATION OF INDEPENDENTLY FAILING COMPONENTS. (DAN 289)

MODULE
A PROGRAM UNIT THAT IS DISCRETE AND IDENTIFIABLE WITH RESPECT TO COMPILING, COMBINING WITH OTHER UNITS AND LOADING. (ANSI-X3) (2) A PROGRAM: (A) CHARACTERIZABLE EXTERNALLY AS PERFORMING A SINGLE OPERATION; AND (B) CHARACTERIZABLE INTERNALLY AS LIMITED IN COMPLEXITY. THE COMPLEXITY OF A MODULE MAY BE MEASURED IN TERMS OF: I) THE DEPTH OF NESTING OF ITS CONTROL STRUCTURES; II) THE TOTAL NUMBER OF ITS CONTROL SEGMENTS (I.E. CONTROL STRUCTURES); AND III) THE TOTAL NUMBER OF ITS OPERATIONS. (ABBOTT) (3) A PORTION OF A COMPUTER PROGRAM WHICH PERFORMS IDENTIFIABLE FUNCTIONS IN A SOMEWHAT AUTONOMOUS MANNER, AND WHICH IS USUALLY CONSTRAINED TO SOME MAXIMUM SIZE. (NASA) (4) MODULES ARE CHARACTERIZED BY LEXICAL BINDING, IDENTIFIABLE

PROPER BOUNDARIES, NAMED ACCESS, AND NAMED REFERENCE. THE WORD "MODULE" MAY APPLY TO A SUBPROGRAM, SUBROUTINE, ROUTINE, PROGRAM, MACRO, OR FUNCTION. A "COMPILE MODULE" IS A MODULE OR SET OF MODULES THAT ARE DISCRETE AND IDENTIFIABLE WITH RESPECT TO COMPILING, COMBINING WITH OTHER UNITS, AND LOADING. (DAN 1153) SEE ALSO: PROGRAM MODULE

MODULE ANALYSIS
INDEXING TERM. MAY REFER TO ANALYSIS BEFORE IMPLEMENTATION AS PART OF THE DESIGN OR REQUIREMENTS PHASES OR TO ANALYSIS PERFORMED TO EXTRACT INFORMATION ABOUT VARIOUS ASPECTS OF THE MODULE DURING TESTING. THE ANALYSIS MAY BE MANUAL OR AUTOMATED.

MODULE SIZING
DETERMINATION OF THE OPTIMUM MODULE SIZE TO MINIMIZE COST AND MAXIMIZE RELIABILITY, PROGRAMMER PRODUCTIVITY, COMPILER COST, ETC. (DAN 338)

MODULE TEST
TEST OF A SINGLE MODULE (SEL)

MODULE TESTING
THE INTENT OF THE MODULE OR UNIT TEST IS TO FIND DISCREPANCIES BETWEEN THE MODULE'S LOGIC AND INTERFACES, AND ITS MODULE EXTERNAL SPECIFICATIONS. (THE DESCRIPTION OF THE MODULE'S FUNCTION, INPUTS, OUTPUTS, AND EXTERNAL EFFECTS). THE STEP OF COMPILING THE MODULE SHOULD ALSO BE CONSIDERED AS PART OF THE MODULE TEST SINCE THE COMPILER DETECTS MOST SYNTAX ERRORS AND A FEW SEMANTIC OR LOGIC ERRORS. (DAN 286)

MONITOR
A MONITOR DETERMINES WHICH OF TWO OR MORE PROCESSES COMPETING FOR CONTROL IN ORDER TO EXECUTE HAS PRIORITY. IT ALLOWS THAT WHICH HAS PRIORTY TO TAKE CONTROL AND EXECUTE AND PLACES THE OTHER PROCESS(ES) ON A QUEUE TO AWAIT THEIR TURN TO TAKE CONTROL AND EXECUTE (DAN 420) (2) MONITORS MAY BE CONSIDERED AS RESOURCE ALLOCATORS USING THE SHARED VARIABLES TO ADMINISTER THE RESOURCE ALLOCATION POLICY. (DAN 422)

MOVE
TO READ DATA FROM A SOURCE AND TO WRITE THE SAME DATA ELSEWHERE IN A PHYSICAL FORM WHICH MAY DIFFER FROM THAT OF THE SOURCE. A MOVE DIFFERS FROM A COPY IN THAT IT NEED NOT PRESERVE THE SOURCE. (ANSI-X3H1)

MTS (MODULE TESTING SYSTEM)
AN AUTOMATIC SOFTWARE TEST DRIVER MARKETED BY MANAGEMENT SYSTEMS AND PROGRAMMING LTD.

MULTICS
A COMMERCIAL OPERATING SYSTEM WHICH EVOLVED FROM A RESEARCH TIME-SHARING SYSTEM. (A PART OF HONEYWELL)

MULTIPLE PROCESSOR
A COLLECTION OF PROCESSORS. MULTIPLE PROCESSORS ARE OFTEN USED TO EXECUTE CONCURRENT PROCESSES. (ABBOTT) COMPARE WITH MULTIPROCESSOR.

MULTIPLEX
TO INTERLEAVE THE EVENTS OF TWO OR MORE ACTIVITIES. (ANSI-X3HI)

MULTIPROCESSING
SIMULTANEOUS EXECUTION BY TWO OR MORE PROCESSORS. (ANSI-X3H1) (2) A PROGRAM EXECUTION THAT ALLOWS FOR SIMULTANEOUS EXECUTION OF A SHARED COPY OF A CODED ELEMENT BY TWO OR MORE CPU'S. (DAN 1201)

MULTIPROCESSOR
A COMPUTER EMPLOYING TWO OR MORE PROCESSORS OF COMPARABLE CAPACITY UNDER THE INTEGRATED CONTROL OF A SINGLE OPERATING SYSTEM WHEREIN ALL PROCESSORS SHARE COMMON MEMORY AND INPUT/OUTPUT FACILITIES. (NASA)

MULTIPROGRAMMING
A MODE OF OPERATION THAT PROVIDES FOR THE INTERLEAVED EXECUTION OF TWO OR MORE COMPUTER PROGRAMS BY A SINGLE CENTRAL PROCESSING UNIT. (2) PERTAINING TO THE CONCURRENT EXECUTION OF TWO OR MORE COMPUTER PROGRAMS BY A COMPUTER. (ANSI-X3) (3) THE CONCURRENT EXECUTION OF TWO OR MORE FUNCTIONS AS THOUGH EACH FUNCTION OPERATES ALONE. (ANSI-X3H1) (4) A PROGRAM DESIGN THAT ALLOWS SUPPORT OF MANY FUNCTIONS SIMULTANEOUSLY AS THOUGH EACH FUNCTION OPERATES ALONE. (DAN 1201)

MULTI-LEVEL OPERATING CONFIGURATION (OR SYSTEM)
AN OPERATING SYSTEM OR KERNEL THAT LETS PROGRAMS HAVING DIFFERENT LEVELS OF DATA ACCESSIBILITY OPERATE CONCURRENTLY. LINES OF COMMUNICATION BETWEEN THE CONCURRENT RUNNING PROGRAMS AND DATA ARE UNIDIRECTIONAL. A HIGH LEVEL PROGRAM MAY ACCESS DATA FROM A LOWER LEVEL ONE, BUT THE REVERSE IS NOT TRUE.

MULTI-TASKING
THE CONCURRENT EXECUTION OF TWO OR MORE TASKS BY A COMPUTER. (ANSI-X3-HI)

MUSA'S MODEL
SOFTWARE RELIABILITY MODEL DEVELOPED BY JOHN MUSA OF BELL LABORATORIES, WHIPPANY, NJ

MUST
(MULTIPURPOSE USER-ORIENTED SOFTWARE TECHNOLOGY) A NASA PROGRAM WHOSE OBJECTIVE IS TO CUT THE COST OF PRODUCING SOFTWARE BY PROVIDING AN INTEGRATED SYSTEM OF SUPPORT SOFTWARE TOOLS FOR USE THROUGHOUT THE RESEARCH FLIGHT SOFTWARE DEVELOPMENT PROCESS. (DAN 327)

MUTUAL EXCLUSION
REFERS TO MONITORS AND PROCESS QUEUES. ALTHOUGH THE MONITOR IS SHARED AMONG CONCURRENT PROCESSES, THE EXECUTION OF THE MONITOR PROCEDURES AND FUNCTIONS EXCLUDE EACH OTHER IN TIME. MUTUAL EXCLUSION CAN BE IMPLEMENTED FOR MONITORS BY THE USE OF BOOLEAN SEMAPHORES INITIALIZED TO THE VALUE TRUE. (DAN 422)

NAMED MODULE
A MODULE WHICH CAN BE INVOKED BY NAME (NAMED ACCESS) AND WHICH INTERNALLY MAY INVOKE SUBMODULES BY NAME (NAMED REFERENCE). SUCH INVOCATION IN THE FLOWCHARTED DESIGN IS DENOTED BY THE METHOD OF "STRIPING" THE FLOWCHART SYMBOL. (DAN 1153)

NATURAL LANGUAGE
(ISO) A LANGUAGE WHOSE RULES ARE BASED ON CURRENT USAGE WITHOUT BEING EXPLICITLY PRESCRIBED. (ANSI-X3)

NATURAL LANGUAGE PROCESSING
A PART OF THE TREND TOWARD PEOPLE-ORIENTED MAN-COMPUTER INTERFACES, ATTEMPTS TO ALLOW PEOPLE TO USE AN ENGLISH-TYPE LANGUAGE - AS SPOKEN ENGLISH - TO INTERACT WITH THE COMPUTER. (DAN 273)

NATURAL LANGUAGE THEORY
A MEASURE OF SOFTWARE COMPLEXITY WHICH LINKS KNOWN RESULTS FROM NATURAL LANGUAGE AND INFORMATION THEORIES TO SOFTWARE COMPLEXITY QUESTIONS. (DAN 232)

NESTING
THE PRACTICE OF BUILDING AN OBJECT OF SOME SORT IN TERMS OF OTHER OBJECTS OF THE SAME SORT. (ABBOTT) (2) THE RECURSIVE APPLICATION OF THE IMBEDDING OF STRUCTURES (PROCEDURAL OR DATA) INTO A HIERARCHY OF STRUCTURAL LEVELS OF DEFINITION. (DAN 1153) SEE ALSO LEVEL OF NESTING.

NETWORK
CONNECTION OF TWO OR MORE NODES; IN "COMPUTER NETWORK", THE SPECIFIC NODES CONSIST OF COMPUTERS, OR PROCESSING OR COMMUNICATIONS EQUIPMENT.

NONE USED
NO EXPLICIT TECHNIQUE WAS SPECIFIED TO BE USED. (SEL)

NON-DETERMINISM
CONVERSE OF DETERMINISM (ANSI-X3H1)

NON-PROCEDURAL
CONVERSE OF PROCEDURAL (ANSI-X3H1)

NON-PROCEDURAL SPECIFICATION
A SCHEME WHICH ALLOWS THE DEFINTION OF BEHAVIOR WITHOUT THE SPECIFICATION OF AN ALGORITHM FOR ACHIEVING THE BEHAVIOR. (DAN 242)

N-VERSION PROGRAMMING
THE INDEPENDENT GENERATION OF N>2 FUNCTIONALLY EQUIVALENT PROGRAMS FROM THE SAME INITIAL SPECIFICATION. THE N PROGRAMS POSSESS ALL THE NECESSARY ATTRIBUTES FOR CONCURRENT EXECUTION, DURING WHICH COMPARISON VECTORS ARE GENERATED BY THE PROGRAM AT CERTAIN POINTS. (DAN 315)

OBJECT PROGRAM
A COMPUTER PROGRAM EXPRESSED IN MACHINE LANGUAGE, USUALLY THE RESULT OF TRANSLATING A SOURCE PROGRAM BY AN ASSEMBLER OR COMPILER. (NASA)

OBJECT PROGRAM DATA
THE RESULTING FORM OF A SOURCE LANGUAGE PROGRAM AFTER PROCESSING BY A COMPILER OR ASSEMBLER. THEY ARE ALSO CALLED OBJECT MODULES. THE OBJECT PROGRAM IS IN A FORMAT SUITABLE FOR LOADING AND EXECUTION. IT MAY REQUIRE ADDITIONAL PROCESSING BY A LINK LOADER OR LINK EDITOR. (DAN LD7)

OFFLINE PROCESSING
NON-INTERACTIVE PROCESSING (ANSI-X3H1)

ONLINE PROCESSING
INTERACTIVE PROCESSING, USUALLY BETWEEN A HUMAN AND COMPUTER. (ANSI-X3H1)

ON-BOARD PROCESSING
ALL SOFTWARE COMPONENTS THAT ARE BUILT FOR THE PURPOSE OF SATISFYING SOME ON-BOARD PROCESSING NEED FALL INTO THIS CLASS. ALTHOUGH THE COMPONENT MAY BE BUILT AND TESTED ON A COMPUTER WHICH IS NOT THE REAL FLIGHT COMPUTER IT SHOULD BE CLASSIFIED AS 'ON-BOARD' IF THE FINAL DESTINATION IS THE OBC (ON-BOARD COMPUTER). (SEL)

ON-LINE DEBUGGING
SEE INTERACTIVE DEBUG

ON-LINE TESTING
SEE INTERACTIVE DEBUG

OPAL
(OPERATIONAL PERFORMANCE ANALYSIS LANGUAGE) A HIGH LEVEL TEST LANGUAGE DEVELOPED FOR THE ARMY. (DAN 390)

OPEN-ENDED FLEXIBILITY
SYNONOMOUS WITH ADAPTABILITY. (DAN 781)

OPERATING LEVELS
AN OPERATING LEVEL IS AN ENVIRONMENT IN WHICH PARTICULAR SETS OF OPERATIONS HAVE PARTICULAR MEANINGS. (ANSI-X3H1)

OPERATING SYSTEM
A SYSTEM OF ROUTINES AND SERVICES THAT MONITORS, CONTROLS, ALLOCATES, DEALLOCATES, AND MANAGES THE EXECUTION OF APPLICATIONS PROGRAMS AND OTHER SYSTEMS ROUTINES AND THEIR USAGES OF SYSTEM RESOURCES. (DAN 1153) AN OPERATING SYSTEM HAS AS FUNCTIONS: 1) CREATION OF OBJECTS, PROCESSES, FILES, MODULES, SEGMENTS, 2) MANAGEMENT AND SHARING OF FILES, 3) MANAGEMENT OF COMMUNICATIONS THROUGH SEGMENTS OR MAIL BOXES, 4) MEMORY MANAGEMENT, 5) CPU MANAGEMENT, 6) INPUT/OUTPUT MANAGEMENT. (DAN 269)

OPERATING SYSTEM COMMAND AND RESPONSE LANGUAGE
(OSCRL) THE LANGUAGE USED TO EXPRESS COMMANDS THAT INITIATE PARTICULAR ACTIONS OF AN OPERATING SYSTEM, AND TO EXPRESS RESPONSES CONVEYED BY THE OPERATING SYSTEM. (ANSI-X3H1)

OPERATING SYSTEM DESIGN
THE PROCESS OF DESIGNING AN OPERATING SYSTEM.

OPERATION
A FUNCTION WHICH TRANSFORMS DATA OBJECTS FROM INPUT DOMAIN(S) INTO DATA OBJECTS IN THE OPERATION'S OUTPUT DOMAIN (S). THE INPUT AND OUTPUT DOMAIN(S) OF AN OPERATION ARE THE DATA TYPES OVER WHICH THE OPERATION IS DEFINED. AN OPERATION ON ONE LEVEL OF ABSTRACTION MAY BE DEFINED BY AN ALGORITHM IN TERMS OF OPERATIONS ON A LOWER LEVEL OF ABSTRACTION. (ABBOTT)

OPERATIONAL
THE STATUS GIVEN A SOFTWARE PACKAGE ONCE IT HAS COMPLETED CONTRACTOR TESTING AND IT IS TURNED OVER TO THE EVENTUAL USER FOR USE IN THE APPLICATIONS ENVIRONMENT. (DAN 21)

OPERATIONAL ENVIRONMENT

THE SET OF ALL EXTERNAL STIMULUS AND DATA SOURCES WITH WHICH A SOFTWARE SYSTEM INTERFACES AND COMMUNICATES. (DAN 1201) SEE ALSO ENVIRONMENT.

OPERATIONAL EVALUATION
THE ANALYSIS OF A SYSTEM OPERATING IN ITS' REAL LIFE ENVIRONMENT. (DAN 1201)

OPERATIONAL RELIABILITY
THE RELIABILITY OF THE PROGRAM-AS-IT-PERFORMS AS OPPOSED TO THE RELIABILITY OF THE PROGRAM-AS-IT-IS. (DAN 245)

OPERATIONAL SOFTWARE
THE PORTION OF A DFCAS COMPUTER PROGRAM, INCLUDING REAL TIME EXECUTIVE AND APPLICATION SOFTWARE, WHICH IS DIRECTLY INVOLVED IN FLIGHT CONTROL AND AVIONICS PROCESSING FUNCTIONS. (NASA)

OPERATIONAL TESTING
PERFORMING TESTS ON SOFTWARE IN ITS NORMAL OPERATING ENVIRONMENT. (DAN 1201)

OPERATOR
(1) AGENT PERFORMING AN ACTION. (2) SPECIFICALLY, OPERATOR MAY REFER TO A HUMAN WHO INTERFACES THE ACTIONS NECESSARY TO KEEP THE COMPUTER COMPLEX OPERATING BETWEEN THE USER OR USER INPUT AND THE MACHINE. (3) AN ABSTRACT MACHINE WITH AN ONGOING MACHINE CYCLE. SEE ALSO: OPERATION

OPTIMIZATION
CHANGES IN THE SOURCE CODE TO IMPROVE PROGRAM PERFORMANCE - E.G. RUN FASTER OR USE LESS SPACE. OPTIMIZATION CHANGES ARE NOT ERROR CORRECTION; HOWEVER, IF A CHANGE IS MADE TO USE LESS SPACE TO CONFORM TO THE SPECIFIED SPACE CONSTRAINT, THEN THE TERM "ERROR" APPLIES. (SEL) NOTE: EFFICIENCY IS A QUALITY CHARACTERISTIC; OPTIMIZATION CAN BE A PROCESS WHICH INCREASES EFFICIENCY.

OUTPUT ASSERTION
AN OUTPUT ASSERTION, USUALLY DENOTED BY THE GREEK LETTER PSI, IS A STATEMENT THAT EXPRESSES A RELATION BETWEEN THE INPUT AND OUTPUT VALUES OF A PROGRAM. AN OUTPUT ASSERTION IS USED IN CONJUNCTION WITH AN INPUT ASSERTION TO SPECIFY FORMALLY THE INTENDED FUNCTION OF A PROGRAM. A PROGRAM IS SAID TO BE TOTALLY CORRECT WITH RESPECT TO AN INPUT ASSERTION PHI AND OUTPUT ASSERTION PSI IF IT HALTS SATISFYING PSI ON ALL INPUTS. (SET)

OVERLAY PROGRAM
A COMPUTER PROGRAM THAT ALLOWS SPECIFIC SYSTEM COMPONENTS (LOAD MODULES, CORE, DATA BASE, ETC.) TO BE MODIFIED DURING EXECUTION. IN THE CASE OF MODULES, A PROGRAM WITH AN ERROR CAN BE REPLACED IN CORE WITHOUT BRINGING THE SYSTEM DOWN AND STARTING IT UP AGAIN. SYSTEM PARAMETERS THAT AFFECT PERFORMANCE CAN BE VARIED DURING EXECUTION TO COMPARE VARIOUS PRIORITY, TIMING, ETC. SCHEMES. (DAN 134)

PARAMETER
THE NAME OR VALUE OF INFORMATION TO BE USED. USED IN TWO SENSES IN A PROCEDURE OR MACRO: 1) AS A NAME IN THE DEFINITION (A FORMAL PARAMETER), 2) AS HAVING A SPECIFIC VALUE AS A PARTICULAR INVOCATION (AN ACTUAL PARAMETER). (ANSI-X3H1)

PARANORMAL TERMINATION
UNSTRUCTURED ESCAPES (IN CONTROL) FROM A MODULE IN RESPONSE TO NORMAL EVENTS OR CONDITIONS. MODULES HAVING PARANORMAL TERMINATIONS MAY YET EXHIBIT A FORM OF STRUCTURED CONTROL FLOW, IF PROPERLY CONFIGURED INTO "PARANORMAL EXTENSIONS" OF STRUCTURED PROGRAMMING. (DAN 1153)

PARSE
TO DECOMPOSE A PROGRAMMING UNIT (BLOCK, LINE, PHRASE, WORD) INTO A SET OF ELEMENTARY SUBUNITS (LINES, WORDS, COMMANDS, CHARACTERS).

PARTIAL CORRECTNESS
A PROGRAM IS PARTIALLY CORRECT WITH RESPECT TO AN INTENDED FUNCTION IF, WHEN EXECUTED ON ANY INPUT IN THE DOMAIN OF THAT FUNCTION, IT EITHER TERMINATES RETURNING AS OUTPUT THE VALUE OF THE FUNCTION OR DOES NOT TERMINATE. PARTIAL CORRECTNESS IS WEAKER THAN TOTAL CORRECTNESS, WHICH REQUIRES TERMINATION ON EACH INPUT FOR WHICH THE INTENDED FUNCTION IS DEFINED. (SET) (2) PROGRAM AGREES IN TOTAL WITH ITS FULL SET OF COMPLETE INPUT, OUTPUT, AND INTERMITTENT ASSERTIONS; DIFFERS FROM TOTAL CORRECTNESS IN THAT TERMINATION IS NOT PROVED. SEE ALSO: TOTAL CORRECTNESS.

PARTITIONING
A SOFTWARE ENGINEERING TECHNIQUE WHICH SEGMENTS THE SYSTEM INTO AREAS OF RESPONSIBILITY FOR DESIGN AND DEVELOPMENT. (DAN 301)

PASCAL
A PROGRAMMING LANGUAGE HAVING RELIABILITY AS A MAJOR DESIGN OBJECTIVE.(DAN 389)

PATCH
AN OBJECT CODED PROGRAM CHANGE WRITTEN IN MACHINE CODE, AND INSERTED TO OVERLAY LANGUAGE PRODUCED OBJECT CODE FOR TEMPORARY REPAIR OF A CODING ERROR OR PROGRAM DISCREPANCY UNTIL THE CODED ELEMENT IS REPROCESSED AND REGENERATED AS A SOURCE CODE CHANGE. (DAN 1201)

PATH ANALYSIS
A (1) A SOFTWARE TECHNIQUE WHICH SCANS SOURCE CODE IN ORDER TO DESIGN AN OPTIMAL SET OF TEST CASES TO EXERCISE THE PRIMARY PATHS IN A SOFTWARE MODULE. (DAN 142) (2) A TECHNIQUE WHICH DEFINES A PRACTICAL MEASURABLE MEANS OF DETERMINING AN OPTIMAL NUMBER OF TEST CASES BY EXAMINING SOURCE CODE AND DETERMINING THE MINIMUM SET OF PATHS WHICH EXERCISE ALL LOGICAL BRANCHES OF A PROGRAM. (DAN LD7)

PATH CONDITION
THE COMPOUND CONDITION WHICH MUST BE SATISFIED BY THE INPUT DATA POINT IN ORDER THAT THE CONTROL PATH BE EXECUTED. IT IS THE CONJUNCTION OF THE INDIVIDUAL PREDICATE CONDITIONS WHICH ARE GENERATED AT EACH BRANCH POINT ALONG THE CONTROL PATH. NOT ALL THE CONTROL PATHS THAT EXIST SYNTACTICALLY WITHIN THE PROGRAM ARE EXECUTABLE. IF INPUT DATA EXIST WHICH SATISY THE PATH CONDITION, THE CONTROL PATH IS ALSO AN EXECUTION PATH AND CAN BE USED IN TESTING THE PROGRAM. IF THE PATH CONDITION IS NOT SATISFIED BY ANY INPUT VALUE, THE PATH IS SAID TO BE INFEASIBLE, AND IS OF NO INTEREST IN TESTING THE PROGRAM. (DAN 842)

PATH EXPRESSIONS

A TYPE OF COMMAND PATH CONNECTED TO A FUNCTION WHICH DESCRIBES THE COOPERATION BETWEEN THE SUBFUNCTIONS OF THIS FUNCTION.

PDL

A PROGRAM DESIGN LANGUAGE (OFTEN CALLED PSEUDOCODE). USED IN THE DESIGN AND CODING PHASES OF A PROJECT, PDL IS A LANGUAGE THAT CONTAINS A FIXED SET OF CONTROL STATEMENTS AND A FORMAL OR INFORMAL WAY OF DEFINING AND OPERATING ON DATA STRUCTURES. PDL CODE MAY OR MAY NOT BE MACHINE READABLE, AND FOR THIS STUDY IS NOT CONSIDERED AS DOCUMENTATION, BUT AS AN INTEGRAL PART OF THE FINISHED SOURCE PROGRAM. (SEL)

PEER CODE REVIEW

A PEER CODE REVIEW (PCR) IS A PROCESS BY WHICH A TEAM OF PROGRAMMING PERSONNEL DO AN IN-DEPTH REVIEW OF A PROGRAM OR PORTION OF A PROGRAM BY INSPECTION TO DETECT ERRORS AND IMPROVE PROGRAM RELIABILITY...TYPICALLY, THE RESPONSIBLE PROGRAMMER LEADS HIS TECHNICAL PEERS THROUGH THE LISTING OF THE PROGRAM, EXPLAINING THE FUNCTION OF EACH LINE OF CODE AND ITS ROLE IN THE OVERALL PROGRAM. THE REVIEW PARTICIPANTS, MEANWHILE, ASK QUESTIONS AND MAKE COMMENTS RELEVANT TO POTENTIAL ERRORS. TECHNIQUES, STYLE, ETC. THE PRIMARY REASON FOR PCR'S IS IMPROVING PROGRAM RELIABILITY AND MAINTAINABILITY. BECAUSE THE PCR TEAM EXAMINES THE SUBJECT PROGRAM CLOSELY, FEWER ERRORS SHOULD SLIP THROUGH UNDETECTED. PCR'S HAVE SOME FRINGE BENEFITS. PROGRAMMERS ARE MOTIVATED TO DO A BETTER JOB, KNOWING THAT THEIR WORK WILL BE CRITIQUED BY THEIR PEERS. ADDITIONALLY PROGRAMMERS WILL COLLECTIVELY IMPROVE THEIR TECHNIQUES AND STYLE AS THEY SHARE CONCEPTS AT THE REVIEW. BACKUP KNOWLEDGE IN PROGRAM FUNCTIONING IS ALSO OBTAINED. CODE REVIEW IS ALSO DRUDGERY AND REQUIRES MOTIVATION OF PARTICIPANTS FOR REWARDING PARTICIPATION. AN OFFSHOOT OF THIS TECHNIQUE IS INSPECTION, WHICH INVOLVES THIS SAME PROCESS, BUT WITHOUT DIRECT INTERACTION WITH THE PROGRAMMER...ALSO SEE - CODE VERIFICATION DESK CHECKING, SOFTWARE SNEAK CIRCUIT ANALYSIS, EGOLESS PROGRAMMING, FOREIGN DEBUG. (SET)

PERFORMANCE

THE EVALUATION OF NON LOGICAL PROPERTIES (I.E. COMPUTER RUN TIME, RESOURCE UTILIZATION) OF A SOFTWARE SYSTEM. PERFORMANCE IS MEASURED IN TERMS OF THE AMOUNT OF RESOURCES REQUIRED BY A SOFTWARE SYSTEM TO PRODUCE A RESULT. (DAN 154) (2) A MEASURE OF THE CAPACITY OF AN INDIVIDUAL OR TEAM TO BUILD SOFTWARE CAPABILITIES IN SPECIALIZED OR GENERALIZED CONTEXTS. PERFORMANCE DISTINGUISHES BETWEEN WORK AND EFFORT, AS IT INCLUDES PRODUCTIVITY AS ONE COMPONENT OF ITS MEASURE. HOWEVER, PERFORMANCE ALSO MEASURES QUALITY OF WORK AS MEASURED BY OTHER CRITERIA AS WELL, AS SET FORTH IN A PRIORITIZED LIST OF "COMPETING CHARACTERISTICS" EARLY IN DEVELOPMENT. (DAN 1153)

PERFORMANCE EVALUATION

THE DEGREE TO WHICH A SYSTEM MEETS STIPULATED OR GENERALLY ACCEPTED GOALS. (DAN 434)

PERFORMANCE ORIENTED DATA

INFORMATION REGARDING MANAGEMENT ITEMS. (DAN LD7)

PERFORMANCE REQUIREMENTS

SPECIFICATION OF THE TIME AND SPACE REQUIREMENTS WHICH MUST BE MET. (DAN 141) (2) THE SPECIFIED SET OF GOALS WHICH MUST BE ATTAINED BY A SOFTWARE SYSTEM. (DAN 1201)

PERFORMANCE SPECIFICATION
THE OFFICIAL DOCUMENT CONTAINING THE PERFORMANCE REQUIREMENTS FOR A PROGRAM. (DAN 1201)

PERIPHERAL SIMULATOR
A COMPUTER PROGRAM USED TO TEST CRITICAL COMPUTER/PERIPHERAL INTERFACES THAT EXIST IN REAL-TIME APPLICATIONS. THESE SIMULATORS RANGE FROM FUNCTIONAL IN WHICH CASE THE PERIPHERAL PROVIDES FEEDBACK TO THE PROGRAM ON THE ASSUMPTION THAT ALL INTERFACE CONSTRAINTS ARE SATISFIED; TO HIGH-FIDELITY SIMULATIONS IN WHICH THE INTERFACE CONSTRAINTS MUST BE MODELED, TO A DETAILED LEVEL OF TIMING AND MESSAGE FORMATTING. (DAN 134)

PETRI NETS
A TOOL FOR THE MODELING AND ANALYSIS OF SYSTEMS WITH CONCURRENT EVOLUTION. (DAN 264)

PHASE OF PRODUCTION
THAT WORK RELATED TO THE COMPLETION OF A SPECIFIED SET OF MODULES IN CONFORMANCE WITH REQUIREMENTS AND GOALS. IN TOP-DOWN DEVELOPMENTS, A SET OF MODULES THAT ARE CURRENTLY DUMMY STUBS BECOMES THE NEXT IMPLEMENTATION PHASE. (DAN 1153)

PLAN DATA
DATA DESCRIBING THE METHOD OR SCHEME OF ACTION FOR A PROJECT THAT WILL BE INCLUDED IN THE MANAGEMENT REPORTS. (DAN 137)

PLANNING
A TECHNIQUE THAT INCLUDES THE EVALUATION OF PROJECT REQUIREMENTS IN TERMS SUCH THAT LOGICAL ASSIGNMENTS CAN BE MADE. ALSO SEE *PLANNING AND SCHEDULING*. (DAN LD7)

PLANNING AND SCHEDULING
ALL ACTIVITIES THAT INCLUDE AN EVALUATION OF THE PROJECT'S REQUIREMENTS AND MAKING ASSIGNMENTS TO CONDUCT THE PROJECT. (DAN LD7)

PL/I
A HIGH LEVEL PROGRAMMING LANGUAGE.

POINTERS
A POINTER IS AN IDENTIFIER THAT INDICATES THE LOCATION OF AN ITEM OF DATA. (ANSI-X3)

POISSON PROCESS
INDEXING TERM. REFERS TO THE MATHEMATICAL METHODOLOGY WHICH IS USED TO CONSTRUCT, OR WHICH IS THE FORM ASSUMED BY, A PARTICULAR MODEL.

PORTABILITY
PORTABILITY IS THE PROPERTY OF A SYSTEM WHICH PERMITS IT TO BE MAPPED FROM ONE ENVIRONMENT TO A DIFFERENT ENVIRONMENT. (2) "PORTABILITY" DESIGNATES THE FACT THAT FOR MANY DIFFERENT MACHINES AND OPERATING SYSTEMS, COPIES OF THE PRODUCT CAN BE DELIVERED WITH UNIFORM OPERATING CHARACTERISTICS, FROM THE USER'S POINT OF VIEW, ANY INPUT WHICH IS VALID ON ONE SUPPORTED SYSTEM IS VALID ON ANY OTHER SUPPORTED SYSTEM, AND WILL PRODUCE IDENTICAL OUTPUT. (DAN 283) (3) CODE POSSESSES THE CHARACTERISTIC PORTABILITY TO THE EXTENT THAT IT

CAN BE OPERATED EASILY AND WELL ON COMPUTER CONFIGURATIONS OTHER THAN ITS CURRENT ONE...THIS IMPLIES THAT SPECIAL LANGUAGE FEATURES, NOT EASILY AVAILABLE AT OTHER FACILITIES, ARE NOT USED; OR THAT STANDARD LIBRARY FUNCTIONS AND SUBROUTINES ARE SELECTED FOR UNIVERAL APPLICABILITY, ETC. (DAN 239) (4) PORTABILITY IS THE PROPERTY OF A SYSTEM WHICH ALLOWS IT TO BE MOVED TO THE NEW ENVIRONMENT WITH RELATIVE EASE. (DAN 781)

PRECISION
PRECISION IN SOFTWARE IS A MEASURE OF THE DEGREE TO WHICH ERRORS TEND TO HAVE THE SAME ROOT CAUSE. SOFTWARE PRECISION COULD BE CONSIDERED AS THE RATIO OF SOURCE BUGS TO THE EFFECTS THEY CAUSE. (DAN 781) (2) A MEASURE OF THE DEGREE OF DISCRIMINATION WITH WHICH A QUANTITY CAN BE STATED, AS OPPOSED TO ACCURACY, WHICH STATES THE DEGREE TO WHICH THAT QUANTITY IS FREE FROM ERROR. (DAN 1153)

PRECOMPILER
A PARTICULAR TYPE OF COMPUTER PROGRAM WHICH HAS THE FOLLOWING CHARACTERISTICS: 1, IT IS NORMALLY EXECUTED IMMEDIATELY PRECEDING A PROGRAM COMPILATION. 2. ITS INPUT CONSISTS OF PROGRAMMING STATEMENTS OF WHICH ALL OR PART ARE UNACCEPTABLE TO THE COMPILER. 3. IT GENERATES, AS OUTPUT, A COMPUTER PROGRAM IN A SYNTAX ACCEPTABLE TO THE COMPILER. (DAN 142) (2) A COMPUTER PROGRAM USED TO ADD CAPABILITIES TO A SYSTEM, AS IMPLEMENTED BY A LANGUAGE PROCESSOR, THAT PROVIDES SPECIAL-PURPOSE FEATURES NOT NORMALLY INCLUDED AS PART OF ITS INPUT.

PREDICATE
A LOGICAL PROPOSITION OR ASSERTION CONCERNING THE STATE OF A PROGRAM AT A GIVEN POINT, HAVING EITHER A TRUE OR FALSE VALUE. CONCERNING PROGRAM CORRECTNESS, ALL SUCH ASSERTIONS MUST BE AXIOMS OR BE PROVED TRUE. (DAN 1153)

PRE-EXECUTION TOOLS
TOOLS THAT OPERATE ON THE LINGUISTIC DESCRIPTION OF A PROGRAM AND DO NOT REQUIRE ITS EXECUTION. EXAMPLES INCLUDE: SYNTAX CHECKING, INTERACTIVE COMPILERS, PROGRAM REFERENCE LISTINGS, FLOW CHARTS, AND REFORMATTERS. (DAN LD7)

PREVENTIVE MAINTENANCE
MAINTENANCE SPECIFICALLY INTENDED TO PREVENT FAULTS FROM OCCURRING. CORRECTIVE MAINTENANCE AND PREVENTIVE MAINTENANCE ARE BOTH PERFORMED DURING MAINTENANCE TIME. CONTRAST WITH CORRECTIVE MAINTENANCE. (ANSI-X3)

PREVENTIVE MAINTENANCE TIME
TIME, USUALLY SCHEDULED, USED TO PERFORM PREVENTIVE MAINTENANCE. (ANSI-X3)

PRIORITY
THE RANK ASSIGNED TO AN ENTITY FOR THE PURPOSE OF DETERMINING THE ORDER IN WHICH COMPETING ENTITIES MAY USE SYSTEM RESOURCES. (ANSI-X3H1)

PRIVILEGE(D)
A RIGHT OR IMMUNITY GRANTED AS A PECULIAR BENEFIT TO A PERSON OR PROCESS. USUALLY USED AS A CLASS - A PRIVILEGED PROGRAM CAN USE OR ACCESS SENSITIVE ITEMS NOT GRANTED TO OTHERS. A PRIVILEGED USER MAY ACCESS OR USE SENSITIVE ITEMS NOT GRANTED TO OTHER USERS. (ANSI-X3H1)

PROBLEM REPORT ANALYSIS
INDEXING TERM. REFERS TO THE PROCESS OF EXTRACTING DATA FROM SOFTWARE PROBLEM REPORTS (SPR) AND PROBLEMS RELATED TO THE DIFFICULTY OF COLLECTING AND ANALYZING THE SPR AS WELL AS RELIABILITY CONCERNS ABOUT THE DATA EXTRACTED.

PROCEDURAL SPECIFICATIONS
A SPECIFICATION OF A COMPONENT IN SOME ALGORITHMIC MANNER (E.G., USING PDL OR A FLOW CHART). THE SPECIFICATION SAYS HOW THE PROGRAM IS TO WORK. A SPECIFICATION OF A SOFTWARE COMPONENT IN SOME ALGORITHMIC MANNER (E.G. USING PSL OR A FLOW CHART). THE SPECIFICATION SAYS HOW THE PROGRAM IS TO WORK. CONTRAST WITH FUNCTIONAL SPECIFICATIONS.

PROCEDURE DESIGN LANGUAGE
A LANGUAGE FOR SPECIFYING ALGORITHMS IN ORDINARY ENGLISH OR OTHER LANGUAGE NOT TO BE COMPILED. KEYWORDS USUALLY APPEAR, SO AS TO FORMAT TEXT AND CONFORM THE SPECIFICATIONS INTO A STRUCTURED FORM. ALSO CALLED A "PROGRAM DEFINITION LANGUAGE" (DAN 1153)

PROCEDURE FACILITY
THE CAPABILITY TO USE AND DEFINE PROCEDURES. (ANSI-X3H1)

PROCEDURE(S)
PROCEDURES ARE UNITS OF CODE RUN BY PROCESSES. (DAN 347) (2) (ISO) THE COURSE OF ACTION TAKEN FOR THE SOLUTION OF A PROBLEM... THE DESCRIPTION OF THE COURSE OF ACTION TAKEN FOR THE SOLUTION OF A PROBLEM. (ANSI-X3) (3) A PROCEDURE DEFINITION IS A SEQUENCE OF SOURCE LANGUAGE INSTRUCTIONS. A PROCEDURE CALL IS A TRANSFER OF CONTROL TO THE PROCEDURE DEFINITION. PROCEDURES MA. DEFINE AND USE PARAMETERS WHICH MAY HAVE DEFAULT VALUES. (ANSI-X3H1)

PROCESS
A MODEL REPRESENTATION OF AN INDEPENDENTLY ACTIVE ENTITY IN THE REAL WORLD. (DAN 250) (2) A PROCESS IS A FINITE STATE MACHINE. (DAN 612) (3) AN INTEGRATED ACTIVITY WHICH IS DEFINED BY A PARTICULAR SEQUENCE OF EVENTS AND ENVIRONMENTS. (4) AN ABSTRACT MACHINE WITH AN ONGOING MACHINE CYCLE. (ABBOTT) (5) A SEQUENCE OF OPERATIONS EXECUTED ONE AT A TIME. TWO PROCESSES ARE THEN CONCURRENT IF THEIR OPERATIONS CAN OVERLAP OR INTERLEAVE ARBITRARILY IN TIME. (DAN 1153)

PROCESS CONSTRUCTION
A TECHNIQUE USED TO COMBINE AND LINK INDEPENDENTLY-CODED MODULES INTO A RUN-TIME PROCESS. THESE INCLUDE LINKAGES TO THE OPERATING SYSTEM. THE TECHNIQUE ALLOWS FOR RAPID RECONFIGURATION BASED ON STIMULI FROM THE RUN-TIME ENVIRONMENT OF A SOFTWARE SYSTEM TO REFLECT CHANGES MADE TO A NUMBER OF ITS MODULES. SPECIFIC COMPUTER PROGRAMS ARE AVAILABLE THAT SERVE AS AIDS TO IMPLEMENTATION THESE INCLUDE SPECIAL-PURPOSE EDITORS AND CONTROL PROGRAMS. (DAN 134)

PROCESS DESIGN LANGUAGE - PDL
(1) A FORMAL ALGORITHMIC SPECIFICATION OF A SOFTWARE COMPONENT. (2) AN EXTENSION OF PASCAL DEVELOPED FOR THE BALLISTIC MISSILE DEFENSE ADVANCED TECHNOLOGY CENTER OF THE DEPT. OF DEFENSE BY TEXAS INSTRUMENTS, INC.

PROCESS QUEUES
A QUEUE CONTAINING PROCESSES AWAITING CONTROL OF THE MONITOR SO THAT EXECUTION MAY TAKE PLACE OR A QUEUE CONTAINING PROCESSES AWAITING LOCAL CONDITION VARIABLES TO BECOME TRUE BEFORE THE PROCESS MAY EXECUTE. (DAN 420)

PROCESSOR
A PHYSICALLY BASED ABSTRACT MACHINE. AN ABSTRACT MACHINE HAVING THE PHYSICAL CAPACITY TO PERFORM ITS DEFINED OPERATIONS. (ABBOTT)

PRODUCT
EVERYTHING CONTRACTED FOR, PRODUCED FOR, AND DELIVERED TO THE CUSTOMER. EXAMPLES INCLUDE: HARDWARE, PROGRAMS, DOCUMENTS, AND TRAINING. (DAN LD7)

PRODUCT CERTIFICATION
A DEMONSTRATION THAT THE ACTUAL SYSTEM PERFORMANCE CORRESPONDS TO THE EXPECTED SYSTEM PERFORMANCE. (DAN LD7)

PRODUCT SAFETY
INDEXING TERM. REFERS TO METHODS TO PREVENT, AND CONCERN ABOUT, MALFUNCTIONS IN EQUIPMENT DUE TO DEFICIENCIES IN SOFTWARE WHICH COULD CAUSE INJURY OR DEATH TO HUMAN BEINGS.

PRODUCT SAFETY EVALUATION
INDEXING TERM. REFERS TO A QUANTITATIVE ASSESSMENT OR MEANS OF MAKING A QUANTITATIVE ASSESSMENT OF SOFTWARE PRODUCT SAFETY.

PRODUCT WARRANTY
A PROCESS THAT PRECEDES CONFIGURATION CONTROL AND QUALITY ASSURANCE ACTIVITIES. THIS PROCESS IS A DEVICE FOR CUSTOMER FEEDBACK AFTER SOFTWARE SYSTEM HAS BEEN DELIVERED. IT PROVIDES: 1) A MEANS BY WHICH THE CUSTOMER REPORTS SUSPECTED PROBLEMS; 2) A MEANS FOR TECHNICAL AND CONTRACTUAL EVALUATION OF THESE PROBLEMS; 3) ARBITRATION AND APPEAL PROCEDURES; AND 4) A MEANS TO RE-CERTIFY THE CORRECTED SYSTEM. ALSO SEE PRODUCT WARRANTY AND MAINTENANCE, AND PRODUCT WARRANTY PROCEDURES. (DAN LD7)

PRODUCT WARRANTY AND MAINTENANCE
A DEVICE, INDICATING THE PROCEDURES TO BE FOLLOWED DURING THE WARRANTY PERIOD OF A SYSTEM, FOR CUSTOMER FEEDBACK AFTER THE DELIVERY OF A SYSTEM. (DAN LD7)

PRODUCT WARRANTY PROCEDURES
A CUSTOMER FEEDBACK DEVICE THAT PRECEDES THE DELIVERY OF A PRODUCT, AND INDICATES THE PROCEDURES TO BE FOLLOWED DURING THE WARRANTY PERIOD OF A SYSTEM. (DAN LD7)

PRODUCTION
THAT PORTION OF A SOFTWARE IMPLEMENTATION THAT HAS TO DO WITH THE GENERATION OF CODE AND DOCUMENTATION AND THE CHECKOUT FOR CORRECTNESS BY PRODUCTION PERSONNEL. PRODUCTION PROGRAMMING IS CHARACTERIZED BY THE APPLICATION OF TRADEOFFS, KNOWN ALGORITHMS, AND STATE-OF-THE-ART SOLUTION METHODS TOWARD SOFTWARE GENERATION, AS OPPOSED TO PROGRAMMING PERFORMED TO EXTEND THE CURRENT STATE OF THE ART. (DAN 1153)

PRODUCTION LIBRARIES

A TECHNIQUE USED TO PROVIDE CONSTANTLY UP-TO-DATE REPRESENTATIONS OF THE COMPUTER PROGRAMS AND TEST DATA IN BOTH COMPUTER AND HUMAN READABLE FORMS. THE CURRENT STATUS AND PAST HISTORY OF ALL CODE GENERATED IS ALSO MAINTAINED. SPECIFIC LIBRARY PROGRAMS ARE AVAILABLE TO SERVE AS AIDS TO IMPLEMENTATION. (DAN 134) SEE ALSO DEVELOPMENT SUPPORT LIBRARIAN AND PROGRAM LIBRARY SYSTEMS.

PRODUCTION RUN

THE OPERATION OF A SOFTWARE SYSTEM UNDER REAL OPERATING CONDITIONS AND THE PRODUCTION OF USEFUL PRODUCTS FOR THE CUSTOMER. THIS IS CONTRASTED WITH A TEST RUN, WHICH IS THE OPERATION OF A SOFTWARE SYSTEM TO TEST ITS PERFORMANCE. (DAN LD7)

PRODUCTIVITY

TRADITIONALLY, THE GENERALLY ACCEPTED DESCRIPTION (IF NOT DEFINITION) OF PROGRAMMING PRODUCTIVITY HAS BEEN "LINES-OF-CODE/MAN-MONTH" (I.E. QUANTITY OF CODE PRODUCED). (2) INSTRUCTIONS/STAFF-YEAR FOR EITHER TOTAL RESOURCES EXPENDED IN THE SOFTWARE DEVELOPMENT CYCLE OR REAL-TIME SOFTWARE DEVELOPMENT. (DAN459) (3) PRODUCTIVITY IS THE RATE OF PRODUCTION OF COMPUTER SOFTWARE. THIS RATE IS NORMALLY MEASURED IN THE QUANTITY OF CODE AND DOCUMENTATION PRODUCED. THE DEFINITION OF PRODUCTIVITY MAY CONTAIN AT LEAST THREE ADDITIONAL ELEMENTS: A) A QUALITATIVE ELEMENT CONCERNED WITH THE CORRECTNESS AND EFFICIENCY OF THE SOFTWARE, B) A QUALITATIVE ELEMENT CONCERNED WITH THE DIFFICULTY OF THE APPLICATIONS BEING IMPLEMENTED INCLUDING SIZE AND COMPLEXITY, C) AN ELEMENT CONCERNED WITH THE COST OF PRODUCING THE SOFTWARE. (SET)

PRODUCTIVITY FACTORS

THOSE FACTORS WHICH INFLUENCE THE PRODUCTIVITY RATE SUCH AS, NATURE OF THE SYSTEM, STATE OF THE SOFTWARE DEVELOPMENT PROCESS, QUALITY OF SYSTEMS ENGINEERING, DESIGN, SIZE OF THE PROJECT (INVERSE EFFECT), ETC. (DAN 459)

PROFILE

A COMPENDIUM OF INFORMATION WHICH CONTRIBUTES TO THE DEFINITION OF AN ENVIRONMENT. (ANSI-X3H1)

PROGRAM

A PROGRAM IS A COLLECTION OF OPERATIONS OR ABSTRACT ENTITY DESIGNED TO CAUSE THE COMPUTER EQUIPMENT TO EXECUTE AN OPERATION OR OPERATIONS... COMPUTER PROGRAMS INCLUDE OPERATING SYSTEMS, ASSEMBLERS, COMPILERS, INTERPRETERS, DATA MANAGEMENT SYSTEMS, UTILITY PROGRAMS, AS WELL AS APPLICATIONS PROGRAMS SUCH AS PAYROLL, INVENTORY CONTROL, OPERATIONAL FLIGHT, SATELLITE NAVIGATION, AUTOMATIC TEST, CREW SIMULATOR, AND ENGINEERING ANALYSIS PROGRAMS. COMPUTER PROGRAMS MAY BE EITHER MACHINE-DEPENDENT OR MACHINE-INDEPENDENT, AND MAY BE GENERAL PURPOSE OR BE DESIGNED TO SATISFY THE REQUIREMENTS OF A SPECIALIZED PROCESS OR A PARTICULAR USER. (SET) (2) THE LOWEST LEVEL OF MODULE THAT CAN BE ASSEMBLED OR COMPILED AND CAN BE EXECUTED AS A SINGLE ENTITY. (DAN 137) (3) AN ALGORITHM ALONG WITH A PARTICULAR COLLECTION OF DATA OBJECTS TO WHICH THE ALGORITHM IS APPLIED. A PROGRAM IS TAKEN AS INDEPENDENT OF THE PROGRAMMING LANGUAGE IN WHICH IT IS EXPRESSED. (ABBOTT)

PROGRAM ANALYSIS

THE PROCESS OF GATHERING INFORMATION ABOUT A PROGRAM (E.G., NUMBER OF PATHS,

FREQUENCIES WITH WHICH EACH PATH IS RUN, RUNNING TIME OF EACH PATH, PROBABILITY OF ERROR ALONG EACH PATH, NUMBER OF BRANCHES) AND USING THAT INFORMATION TO EVALUATE COST FUNCTIONS, TO MEASURE RELIABILITY AND PERFORMANCE, AND TO ASSIST IN TESTING AND VERIFICATION PROCEDURES.

PROGRAM ANNOTATION
THE PROCESS OF INVARIANT ASSERTIONS. (DAN 263)

PROGRAM ARCHITECTURE
THE STRUCTURE AND ORGANIZATION OF A COMPUTER PROGRAM AS REFLECTED BY THE RELATIONSHIPS BETWEEN ITS FUNCTIONS AND THE TRANSFER OF CONTROL AMONG THE MODULES PERFORMING THESE FUNCTIONS. (NASA)

PROGRAM COMPLEXITY
PROGRAM COMPLEXITY IS AN INDICATOR OF PROGRAM READABILITY. IT IS A FUNCTION OF THE NUMBER OF EXECUTION PATHS IN THE PROGRAM AND THE DIFFICULTY OF DETERMINING THE PATH FOR AN ARBITRARY SET OF INPUT DATA. (DAN 262)
ADDITIONAL LIST: DAN 314 P142

PROGRAM CONTRACTION
THE PROCESS OF REMOVING UNWANTED AND/OR UNNEEDED CAPABILITY FROM A PROGRAM. (DAN 275)

PROGRAM CORRECTNESS
A CORRECT PROGRAM IS ONE THAT HAS BEEN PROVED TO MEET ITS SPECIFICATIONS. (DAN 237)

PROGRAM DESCRIPTION SPECIFICATIONS
THESE ARE INFORMATIONAL DOCUMENTS PRODUCED FOR THE CUSTOMER, USUALLY ACCORDING TO HIS REQUIREMENTS. (DAN LD7)

PROGRAM DESIGN LANGUAGE
(PDL) A DESIGN TOOL USED TO FACILITATE THE TRANSLATION OF FUNCTIONAL SPECIFICATIONS INTO COMPUTER INSTRUCTIONS. (DAN 14) (2) INTENDED TO BE COMPARABLE TO THE BLUEPRINT IN HARDWARE, PROGRAMMING DESIGN LANGUAGES STRIVE TO COMMUNICATE THE CONCEPT OF THE SOFTWARE DESIGN IN ALL NECESSARY DETAIL, USING A FORMAL OR STRUCTURED VERSION OF ENGLISH, SOMETIMES CALLED PIDGIN ENGLISH OR PSEUDO-CODE. (DAN 227)

PROGRAM DESIGN METHODOLOGIES
INDEXING TERM. REFERS TO A DOCUMENT DESCRIBING OR ADVOCATING A PARTICULAR DESIGN METHODOLOGY, TO A COMPARISON OF TWO OR MORE DESIGN METHODOLOGIES, OR TO A RELATION OF EXPERIENCES RESULTING FROM THE USE OF A PARTICULAR DESIGN METHODOLOGY.

PROGRAM DEVELOPMENT TOOLS
TOOLS THAT TAKE THE COMPUTER-STORED SOURCE PROGRAM INTO AN OBJECT CODE MODULE AND THEN A LOAD MODULE, AND SUBSEQUENTLY EXECUTE THE CODE IN A SPECIFIC TEST ENVIRONMENT. (DAN LD7)

PROGRAM EVALUATION
THE PROCESS OF STUDYING A PROGRAM TO DETERMINE HOW WELL IT FULFILLS ITS DESIGNED PURPOSE. (DAN 1201)

PROGRAM EXTENSION
THE PROCESS OF ADDING NEEDED OR DESIRED CAPABILITY TO A PROGRAM. (DAN 275)

PROGRAM FAMILIES
WE CONSIDER A SET OF PROGRAMS TO BE A PROGRAM FAMILY IF THEY HAVE SO MUCH IN COMMON THAT IT PAYS TO STUDY THEIR COMMON ASPECTS BEFORE LOOKING AT THE ASPECTS THAT DIFFERENTIATE THEM. (DAN 275)

PROGRAM FLOW ANALYZER
A COMPUTER PROGRAM THAT PROVIDES STATISTICS ON SOURCE CODE STATEMENT USAGE AND TIMING DATA ON PROGRAM ELEMENTS DURING TEST CASE EXECUTIONS. (DAN 134)

PROGRAM IMPLEMENTATION
ALL ACTIVITIES ASSOCIATED WITH SOFTWARE MODULE DESIGN, CODING, TESTING AND THE INTEGRATION OF SOFTWARE MODULES INTO A FUNCTIONAL SYSTEM OPERATING ON THE TOTAL, FINAL HARDWARE CONFIGURATION. (DAN LD7)

PROGRAM INSTRUMENTATION
A QUANTITATIVE ASSESSMENT OF HOW THOROUGHLY A PROGRAM IS EXERCISED BY A SET OF TEST CASES. (DAN LD7)

PROGRAM INSTRUMENTATION TOOLS
TOOLS THAT PROVIDE A MECHANISM FOR MONITORING AND RECORDING INFORMATION ABOUT AN OBJECT PROGRAM AS IT OPERATES. EXAMPLES INCLUDE TRACES AND SNAPSHOT DUMPS. (DAN LD7)

PROGRAM LIBRARY
A CONTROLLED SET OF ALL DOCUMENTATION, BASELINE PROGRAMS, NEW PROGRAMMED ELEMENTS, SUPPORT SOFTWARE AND TOOLS AVAILABLE FOR DEVELOPMENT AND CONFIGURATION MANAGEMENT OF A PROGRAM. (DAN 1201)

PROGRAM LIBRARY SYSTEM
AN AUTOMATED PROGRAM LIBRARY OR SUPPORT SYSTEM WHICH STORES ALL OF THE PROJECT'S WORK PRODUCTS (E.G., SOURCE CODE, OBJECT CODE, TEST CASES, DOCUMENTATION) IN AN INTEGRATED DATA BASE. THE SYSTEM MAY OPERATE IN BATCH OR INTERACTIVE MODE, OR BOTH. IT USUALLY WILL PROVIDE FOR MULTIPLE VERSIONS OF A FILE, WITH ONE VERSION OF EACH FILE TAGGED AS THE PRODUCTION VERSION, AND WITH SECURITY MECHANISMS TO PREVENT UNAUTHORIZED CHANGES TO THE PRODUCTION VERSION. FILES ARE USUALLY PROVIDED FOR SOURCE CODE, OBJECT CODE, JCL, AND PROGRAM DATA. THE PROGRAM LIBRARY SYSTEM CAN ALSO ENCOMPASS REPORT GENERATORS TO ANALYZE DATA FROM THE INTEGRATED DATA BASE TO PROVIDE FOR PROJECT VISIBILITY AND FOR AUDITING PURPOSES. (DAN 286)

PROGRAM LISTING
THE SEQUENCE OF INSTRUCTIONS COMPRISING A COMPUTER PROGRAM, USUALLY IN THE FORM OF A PRINTOUT. (NASA)

PROGRAM MANAGEMENT DIRECTIVE
THE OFFICIAL HQ USAF MANAGEMENT DIRECTIVE USED TO PROVIDE DIRECTION TO THE IMPLEMENTING AND PARTICIPATING COMMANDS AND SATISFY DOCUMENTATION REQUIREMENTS IT WILL BE USED DURING THE ENTIRE ACQUISTION CYCLE TO STATE REQUIREMENTS AND REQUEST STUDIES AS WELL AS INITIATE, APPROVE, CHARGE, TRANSITION, MODIFY, OR TERMINATE PROGRAMS. THE CONTENT OF THE PMD, INCLUDING THE REQUIRED HQ USAF REVIEW AND APPROVAL ACTIONS, IS TAILORED TO THE NEEDS

OF EACH INDIVIDUAL PROGRAM. (AFR800-2)

PROGRAM MANAGEMENT PLAN
THE DOCUMENT DEVELOPED AND ISSUED BY THE PROGRAM MANAGER THAT SHOWS THE INTEGRATED TIME-PHASED TASKS AND RESOURCES REQUIRED TO COMPLETE THE TASK SPECIFIED IN THE PROGRAM MANAGEMENT DIRECTIVE (PMD). (AFR800-2)

PROGRAM MANAGER
THE GENERIC TERM USED TO DENOTE A SINGLE AIR FORCE MANAGER (SYSTEM PROGRAM DIRECTOR, PROGRAM/PROJECT MANAGER, OR SYSTEM/ITEM MANAGER) DURING ANY SPECIFIC PHASE OF THE ACQUISITION LIFE CYCLE. (AFR800-2)

PROGRAM MODIFICATION
INDEXING TERM. REFERS TO CHANGES MADE TO PROGRAMS DURING THE MAINTENANCE PHASE OF THE SOFTWARE LIFE CYCLE.

PROGRAM MODULE
A PROGRAM MODULE IS A DISCRETE, IDENTIFIABLE SET OF INSTRUCTIONS USUALLY HANDLED AS A UNIT BY AN ASSEMBLER, A COMPILER, A LINKAGE EDITOR, A LOADING ROUTINE, OR OTHER TYPE OF ROUTINE OR "SUBROUTINE". (SET) SEE ALSO: MODULE, MODULARITY, MODULAR DECOMPOSITION, ROUTINE, SUBROUTINE.

PROGRAM MUTATION
A TECHNIQUE FOR CREATING HIGH QUALITY TEST DATA. THE APPROACH IS BASED ON THE COMPETENT PROGRAMMER ASSUMPTION; THAT AFTER THE PROGRAMMER HAS COMPLETED HIS JOB, THE PROGRAM IS EITHER CORRECT OR "ALMOST" CORRECT IN THAT IT DIFFERS FROM A CORRECT PROGRAM IN ONLY SIMPLE WAYS, AND IS THUS A MUTANT OF A CORRECT PROGRAM. THE CENTRAL IDEA OF PROGRAM MUTATION IS THE CONSTRUCTION OF A SET OF MUTANTS OF THE TARGET PROGRAM. A MUTANT IS A COPY OF THE TARGET PROGRAM WHICH DIFFERS ONLY BY A SINGLE "MUTATION". A MUTATION IS A TRANSFORMATION OF A PROGRAM STATEMENT IN A WAY WHICH STIMULATES TYPICAL PROGRAM ERRORS. SOME MUTANTS MAY TURN OUT TO BE EQUIVALENT, FUNCTIONALLY, TO THE TARGET PROGRAM. THE REMAINDER SHOULD BE DISTINGUISHED FROM THE TARGET PROGRAM BY SUFFICIENTLY POWERFUL TEST DATA. TEST DATA WHICH IS ABLE TO DISTINGUISH ALL NON-EQUIVALENT MUTANTS OF A TARGET PROGRAM MUST THOROUGHLY EXERCISE THE PROGRAM AND, HENCE, PROVIDE STRONG EVIDENCE OF THE PROGRAM'S CORRECTNESS. (DAN 841,843)

PROGRAM OFFICE
THE FIELD OFFICE ORGANIZED BY THE PROGRAM MANAGER TO ASSIST HIM IN ACCOMPLISHING THE PROGRAM TASKS.

PROGRAM PRODUCTION TOOL
AN ESTABLISHED PROCEDURE OR COMPUTER PROGRAM THAT PROVIDES ASSISTANCE IN THE DEVELOPMENT, IMPLEMENTATION, AND TESTING OF A SOFTWARE SYSTEM. (DAN LD7)

PROGRAM PROTECTION
THE MECHANISMS IMPLEMENTED IN A SYSTEM WHICH PREVENT UNAUTHORIZED ACCESS TO THOSE PARTS OF A SYSTEM (E.G., FILES, DATA STRUCTURES) BELONGING TO A PARTICULAR USER. ALSO, THE TERM CAN APPLY TO THE DEGREE OF PROTECTION PROVIDED BY A SYSTEM FOR A PARTICULAR USER OR CLASS OF USERS.

PROGRAM REFERENCES
SPECIAL LISTINGS, PRODUCED BY MANY COMPILERS, THAT INCREASE THE USER'S

UNDERSTANDING OF THE NATURE OF THE PROGRAM PRODUCED. EXAMPLES INCLUDE: DICTIONARY LISTINGS, CORE-STORAGE MAPS, AND CROSS-REFERENCE LISTINGS. (DAN LD7)

PROGRAM SEGMENT
THE SMALLEST CODED UNIT OF A PROGRAM WHICH CAN BE LOADED AS ONE LOGICAL ENTITY. (DAN 1201) (2) A COMBINATION OF PROGRAM STEPS AND CALLS TO LOWER-LEVEL PROGRAM SEGMENTS. (DAN LD7) (3) A COLLECTED SEQUENCE OF PROGRAM STEPS.

PROGRAM SEQUENCER
A COMPUTER PROGRAM USED TO FORCE EXECUTION OF ALL POSSIBLE PROGRAM INSTRUCTIONS AND BRANCHES TO DETERMINE PROGRAM FLOW, EXECUTE SELDOM-USED BRANCHES, AND TO VERIFY PROPER PROGRAM OPERATIONS. THE AID IS OFTEN USED WITH AN INSTRUCTION SIMULATOR. (DAN 134)

PROGRAM STUB
A TEMPORARY (I.E. DUMMY) UNIT OF SOURCE CODE WHICH IS PART OF AN INCOMPLETE STRUCTURED PROGRAM AND WILL BE REPLACED BY THE ACTUAL UNIT OF CODE WHEN IT IS COMPLETED. (DAN 137) (2) DUMMY PROGRAM SEGMENTS CONTAINING ENOUGH CODE TO ESTABLISH LINKAGE WITH A HIGHER-LEVEL SEGMENT. (DAN LD7)

PROGRAM STUB SIMULATORS
GENERALIZED SUBROUTINES USED IN PROGRAM STUBS THAT SUPPLY THE CODE REQUIRED TO ESTABLISH THE DESIRED LINKAGE WITH THE HIGHER-LEVEL PROGRAM SEGMENTS. THEY INCLUDE GENERALIZED TABLE LOOKUP ROUTINES, RANDOM NUMBER GENERATORS, OR ROUTINES THAT ONLY RECORD THE INVOCATION OF A PROGRAM SEGMENT. (DAN LD7)

PROGRAM SUPPORT LIBRARY
A SOFTWARE SYSTEM WHICH PROVIDES TOOLS TO ORGANIZE, IMPLEMENT AND CONTROL SOFTWARE DEVELOPMENT. SEE ALSO: SOFTWARE DEVELOPMENT LIBRARY, PROGRAMMING SUPPORT LIBRARY

PROGRAM SYNTHESIS
USE OF PROGRAM VERIFICATION TECHNIQUES AND PRINCIPLES IN THE CONSTRUCTION OF PROGRAMS. (DAN 265)

PROGRAM TESTING
SEE TESTING

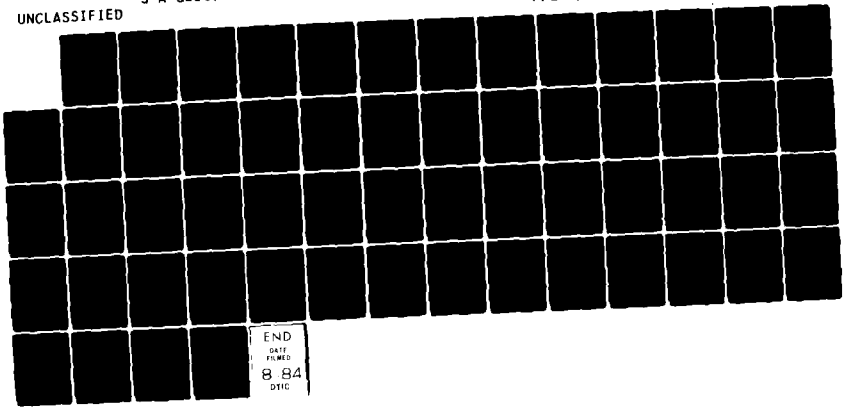
PROGRAM TRANSFORMATIONS
TO REPLACE ONE SEGMENT OF A PROGRAM DESCRIPTION BY ANOTHER, EQUIVALENT DESCRIPTION. (DAN 265) ALSO DAN 250.

PROGRAM UNDERSTANDING
INDEXING TERM. INDICATES THE DEGREE TO WHICH A PROGRAM IS COMPREHENSIBLE AND CAN BE UNDERSTOOD IN FUNCTION AND SCOPE (OTHER THAN BY THE PROGRAMMER WHO WROTE IT), AND ALSO INDICATES THAT THE DOCUMENT DISCUSSES FACTORS WHICH CONTRIBUTE TO PROGRAM UNDERSTANDING.

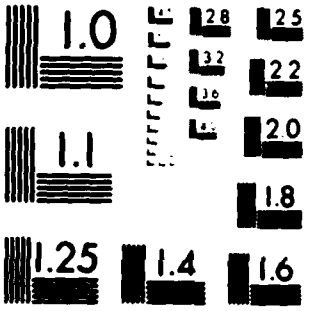
PROGRAM VALIDATION
ALL TECHNIQUES USED TO ENSURE CORRECT PROGRAMS INCLUDING SYSTEM AND SUBSYSTEM TESTS AND SYSTEM INTEGRATION TESTING. (DAN LD7)

AD-A127 689 A BIBLIOGRAPHY OF SOFTWARE ENGINEERING TERMS(U) DATA AND ANALYSIS CENTER FOR SOFTWARE GRIFFISS AFB NY S A GLOSS-SOLER OCT 79 DACS-GLOS-1 F30602-78-C-0255 F/G 5/2 NL

UNCLASSIFIED



END
DATE FILMED
8 84
DTIC


1.0
1.1
1.25
1.4
1.6
2.0
2.2
2.5
1.8

PROGRAM VALIDATION TOOLS
TOOLS THAT ARE USED IN THE PREEXECUTION AND RUN TIME PHASES OF PROGRAM IMPLEMENTATION AND PROGRAM VALIDATION. (DAN 107)

PROGRAMMER
A PROGRAMMER IS A PERSON WHO PRODUCES COMPUTER PROGRAMS. A SENIOR LEVEL PROGRAMMER IS NORMALLY CAPABLE OF PERFORMING ALL SOFTWARE DEVELOPMENT ACTIVITIES INCLUDING DESIGN, CODE, TEST, AND DOCUMENTATION. THE ACTIVITIES OF A MORE JUNIOR LEVEL PROGRAMMER MAY BE LIMITED TO CODING, TEST CASE PREPARATION, AND/OR ASSISTING IN THE MODIFICATION OF EXISTING PROGRAMS AND DOCUMENTATION. ALSO SEE - PROGRAM. (SET)

PROGRAMMER PRODUCTIVITY
THE NUMBER OF VALID SOURCE STATEMENTS CODED PER BUSY HOUR, WHERE VALID SOURCE STATEMENTS ARE THOSE SOURCE STATEMENTS OF A WORKABLE COMPUTER SOURCE PROGRAM. (DAN 314)

PROGRAMMERS APPRENTICE
A COMPUTER SYSTEM WHICH WHEN GIVEN KNOWLEDGE OF BASIC PROGRAMMING TECHNIQUES AND THE ABILITY TO ASSIMILATE APPLICATION DOMAIN CONCEPTS CAN UNDERSTAND A USER'S PROGRAM AND COOPERATE WITH THE USER IN THE DESIGN, IMPLEMENTATION, AND MAINTENANCE OF THE PROGRAM. AN APPRENTICE NEED NOT BE CAPABLE OF PROGRAMMING BY ITSELF BUT CAN AID THE EXPERT PROGRAMMER BY CHECKING HIS WORK IN VARIOUS WAYS. (DAN 720)

PROGRAMMING
PROGRAMMING IS THE ACTIVITY OF DESIGNING, WRITING, AND TESTING IN A COMPUTER LANGUAGE TO ACCOMPLISH A GIVEN TASK. (SET) (2) PRODUCTION OF THE CODE WHICH WILL CONTROL THE SYSTEM AND PERFORM ALL REQUIRED LOGIC AND COMPUTATION. (DAN 773) (3) CODING AND DOCUMENTING A DESIRED SOFTWARE FUNCTION TO COMMUNICATE IT TO A PARTICULAR COMPUTER AND TRAINED PERSONNEL, RESPECTIVELY. (NASA)

PROGRAMMING AIDS
INDEXING TERM. REFERS TO TOOLS, TECHNIQUES, AND PROCEDURES WHICH ARE OF USE IN WRITING PROGRAMS.

PROGRAMMING LANGUAGE
A FORMAL NOTATION IN WHICH PROGRAMS ARE EXPRESSED. (ABBOTT) A FORMAL LANGUAGE COMPOSED OF A REPERTOIRE OF INSTRUCTIONS AND STATEMENTS, HAVING FORMAL SYNTAX AND LEXICAL RULES, USABLE IN COMPOSING COMPUTER PROGRAMS WHICH REQUIRE TRANSLATION TO BE MACHINE EXECUTABLE. (NASA)

PROGRAMMING LANGUAGE COMPLEXITY
PROGRAMMING LANGUAGE COMPLEXITY IS AN INDICATOR OF THE READABILITY OR UNDERSTANDABILITY OF A PROGRAMMING LANGUAGE. (DAN 297) ALSO RELATED TO PROGRAM LENGTH.

PROGRAMMING SPECIFICATION
THAT PORTION OF THE SOFTWARE SPECIFICATION DOCUMENT WHICH SETS FORTH DESCRIPTIONS OF ALGORITHMS, DATA STRUCTURES, THE MODULAR DEFINITION, ETC., IN SUFFICIENT DETAIL THAT THE PROGRAM CAN BE CODED WITHOUT FUNCTIONAL OR ALGORITHMIC AMBIGUITY. (DAN 1153)

PROGRAMMING STANDARDS

SEE STANDARDS, SOFTWARE ENGINEERING STANDARDS

PROGRAMMING SUPPORT LIBRARY
(PSL) A REPOSITORY FOR DATA NECESSARY FOR THE ORDERLY DEVELOPMENT OF COMPUTER PROGRAMS USING STRUCTURED PROGRAMMING TECHNOLOGY. THE DATA REPOSITORY IS IN TWO FORMS: DATA IS STORED IN MACHINE READABLE FORM ACCESSIBLE BY THE COMPUTER AND THE IDENTICAL DATA IS STORED IN HARD COPY FORM IN PROJECT NOTEBOOKS. A PSL ALSO INCLUDES THE NECESSARY COMPUTER AND OFFICE PROCEDURES FOR MANIPULATING THIS DATA. (DAN 140) SEE ALSO: SOFTWARE DEVELOPMENT LIBRARY, PROGRAM SUPPORT LIBRARY

PROGRAMMING TECHNIQUES
INDEXING TERM. METHODS OR MEANS USED TO DEVELOP, DESIGN, OR WRITE A PROGRAM.

PROJECT
A GENERAL TERM USED TO DESCRIBE A SOFTWARE DEVELOPMENT EFFORT. (DAN 137) (2) ALL PROCESSES NECESSARY TO PRODUCE A PRODUCT. (DAN 107)

PROJECT CONSTRUCT DATA
INFORMATION ON DESIGN AND IMPLEMENTATION DETAILS. (DAN 107)

PROJECT DATA BASE
A PROJECT-SPECIFIC CATALOG CONTAINING MANAGEMENT DATA AND PROGRAM DATA SUPPLIED AND USED BY VARIOUS FACILITIES. EXAMPLES INCLUDE: THE CONTENTS NAMES, AND LINKAGES OF ALL THE MODULES IN A PARTICULAR SYSTEM. THE MANAGMENT DATA PERTAINING TO THE TASKS AND MILESTONES OF A SPECIFIC PROJECT. (DAN 107)

PROJECT MANAGEMENT SURVEY
INDICATES THAT A SURVEY WAS DONE ON SOFTWARE ENGINEERING PROJECT MANAGEMENT (INDEXING TERM ONLY)

PROJECT NOTEBOOK
DESIGN EFFORTS INEVITABLY PRODUCE MUCH WRITTEN MATERIAL - MEMORANDA, EXPLANATIONS, REPORTS. THE PROJECT WORKBOOK IS A TOOL OR AID USED TO CAPTURE AND ORGANIZE THESE MATERIALS, SO AS TO BE SURE THESE MATERIALS REACH ALL WHO NEED TO USE THEM, AND ARE AVAILABLE FOR USE LATER. (DAN 227-MODIFIED)

PROJECT PLAN
INTERNALLY CONTROLLED FORMAL DOCUMENT REQUIRED ON ALL PROJECTS WHICH DEFINES TO COMPANY MANAGEMENT THE CONTRACT FORM, THE STATEMENT OF WORK, THE MILESTONES SCHEDULE, THE METHOD OF COMPLETING ALL DELIVERED ITEMS, EVALUATION OF RISK, AND CUSTOMER RELATIONS. (DAN 1201)

PROMPT
TO INFORM A USER THAT A SYSTEM OR PROCESS IS READY FOR THE NEXT COMMAND, DATA ELEMENT, OR OTHER INPUT. BY OFFERING VERBAL OR PICTORIAL SUGGESTIONS, TO ASSIST A USER TO COMPLETE OR CORRECT A COMMAND OR OTHER EXPECTED MESSAGE. (ANSI-X3H1)

PROOF OF CORRECTNESS
A PROOF OF CORRECTNESS IS A STATEMENT OF ASSERTIONS ABOUT A PROGRAM THAT IS VERIFIED BY ANALYTIC METHODS... AN ALTERNATIVE TO EXECUTING TESTS ON SOFTWARE TO DEMONSTRATE ITS CORRECTNESS IS THE METHOD OF ANALYTIC PROOFS. THE VERIFICATION PROCESS CONSISTS OF MAKING ASSERTIONS DESCRIBING THE STATE

OF A PROGRAM, INITIALLY, AT INTERMEDIATE POINTS IN THE PROGRAM FLOW AND AT TERMINATION; AND THEN PROVING THAT EACH ASSERTION IS IMPLIED BY THE INITIAL OR PRIOR ONE AND BY THE TRANSFORMATIONS PERFORMED BY THE PROGRAM BETWEEN EACH TWO CONSECUTIVE ASSERTIONS. AN ASSERTION CONSISTS OF A DEFINITION OF THE RELATIONSHIPS AMONG THE VARIABLES AT THAT POINT IN THE PROGRAM WHERE THE ASSERTION IS MADE. THE PROOFS EMPLOY STANDARD TECHNIQUES FOR PROVING THEOREMS IN THE FIRST-ORDER PREDICATE CALCULUS. PROOF OF THE CORRECTNESS OF A PROGRAM USING THIS APPROACH LESSENS THE NEED FOR EXECUTING TEST CASES, SINCE ALL POSSIBILITIES ARE COVERED BY THE PROOFS. ALSO SEE - COMPUTER PROGRAM VERIFICATION. (SET) (2) AN AGREEMENT, IN TOTAL, OF A PROGRAM WITH ITS ASSERTIONS; ALSO, THE USUAL ADDITIONAL ASSUMPTION IS THAT PROGRAM TERMINATION IS, LIKEWISE, PROVED.

PROOF TECHNIQUE
A METHOD FOR FORMALLY DEMONSTRATING THAT A PIECE OF SOFTWARE PERFORMS ACCORDING TO ITS SPECIFICATIONS. PROOF TECHNIQUES USUALLY USE SOME FORM OF MATHEMATICAL NOTATION TO DESCRIBE THE RESULT OF EXECUTING A PROGRAM. (SEL) SEE ALSO CORRECTNESS PROOFS.

PROPER PROGRAM
A PROGRAM OR PROGRAM SEGMENT, SUCH AS A SUBROUTINE, SUBPROGRAM, OR FUNCTION, WHICH HAS BUT ONE POINT OF ENTRY (IN CONTROL) AND BUT ONE MODE OF EXIT (ALTHOUGH, IF A SUBROUTINE, IT MAY BE CALLED FROM, AND RETURN TO, MANY POINTS IN A PROGRAM). (DAN 1153)

PROTECTION
AN ARRANGEMENT FOR RESTRICTING ACCESS TO OR USE OF A SYSTEM OR PART OF A SYSTEM. (ANSI-X3)

PROTECTION MEASUREMENT
A MEASURE OF THE DEGREE OF PROTECTION PROVIDED FOR A PROGRAM OR SYSTEM.

PROTOCOL
GIVEN A SET OF ENTITIES, A PREDEFINED SET OF RULES THAT CONTROL THE COMMUNICATION AMONG THE ENTITIES IS CALLED A COMMUNICATION PROTOCOL OF THE SET. THE ENTITIES CAN BE PROCESSES OR APPLICATION PROGRAMS IN THE SAME COMPUTER OR IN DIFFERENT COMPUTERS WITHIN A COMPUTER NETWORK. (DAN 451) (2) A PROTOCOL IS A SET OF RULES (FORMULAS OR STANDARDS) ESTABLISHED TO REGULATE THE INTERACTIONS BETWEEN THE ATTACHED ENTRIES IN A COMPUTER NETWORK AND TO ENSURE THAT THEY PROCEED IN AN ORDERLY FASHION. (3) A RULE PRESCRIBING THE INTERFACE DISCIPLINES AND CORRECT PROCEDURES FOR COMMUNICATIONS WITH A PROGRAM, SUBROUTINE, OPERATING SYSTEM, OR HARDWARE DEVICE. (DAN 1153)

PSL JOB
ALL OF THE COMPUTER PROCESSING THAT RESULTS FROM A SINGLE USER REQUEST TO EXECUTE THE PSL FOR THE PURPOSE OF PERFORMING ONE OR MORE PSL FUNCTIONS SUCH AS UPDATE, COMPILE OR OUTPUT. (PSL IS PROGRAM SUPPORT LIBRARY.)

QUALIFICATION TESTING
QUALIFICATION TESTING OF SOFTWARE CONSISTS OF PERFORMING A CONTROLLED EXECUTION OF A DELIVERABLE PROGRAM PACKAGE SUCH THAT ALL SPECIFIED REALTIME AND FUNCTIONAL REQUIREMENTS ARE KNOWN TO BE SATISFIED BY THE PROGRAM PACKAGE. NORMALLY, THIS INVOLVES: DOCUMENTATION DESCRIBING THE TEST (TEST PLAN AND TEST PROCEDURES) APPROVED BY THE CUSTOMER THROUGH A REVIEW PROCESS;

GENERATION OF A TEST SYSTEM (HARDWARE AND SOFTWARE) TO PROVIDE THE SIMULATED ENVIRONMENT, TEST CONTROLLER AND DATA RECORDER; AND CUSTOMER OBSERVED OPERATION OF THE TEST AND COMPILATION OF THE TEST RESULTS. (DAN 1201)

QUALITY
THE DEGREE TO WHICH SOFTWARE CONFORMS TO QUALITY CRITERIA. QUALITY CRITERIA INCLUDE, BUT ARE NOT LIMITED TO, CORRECTNESS, RELIABILITY, VALIDITY, RESILIENCE, USEABILITY, CLARITY, MAINTAINABILITY, MODIFIABILITY, GENERALITY, PORTABILITY, TESTABILITY, EFFICIENCY, ECONOMY, INTEGRITY, DOCUMENTATION, UNDERSTANDABILITY, FLEXIBILITY, INTEROPERABILITY, MODULARITY, REUSABILITY. (THESE ALSO HAVE SUBGROUPS; SEE P2 - 7, VOL 1, DAN 283)

QUALITY ASSURANCE
A PLANNED AND SYSTEMATIC PATTERN OF ALL ACTION NECESSARY TO PROVIDE ADEQUATE CONFIDENCE THAT THE ITEM OR PRODUCT CONFORMS TO ESTABLISHED TECHNICAL REQUIREMENTS. (P730/D5) (2) THE PROCESS OF ACTIVITY DURING WHICH THE SYSTEM DESIGN IS AUDITED TO DETERMINE WHETHER OR NOT IT REPRESENTS A VERIFIABLE AND CERTIFIABLE SPECIFICATION, AND DURING WHICH TEST PLANS AND TEST PROCEDURES ARE FORMULATED AND IMPLEMENTED. THIS ACTIVITY ENSURES THE TECHNICAL COMPLIANCE OF THE SOFTWARE SYSTEM--A PRODUCT--TO ITS REQUIREMENTS AND DESIGN SPECIFICATIONS. QUALITY ASSURANCE IS AN INDEPENDENT AUDIT REVIEW OF ALL PRODUCTS TO ENSURE THEIR COMPLIANCE TO A MANAGEMENT-DIRECTED STANDARD OF QUALITY. (DAN LD7) (3) GUARANTEE MADE BY THE DEVELOPER TO THE CUSTOMER THAT THE SOFTWARE MEETS MINIMUM LEVELS OF ACCEPTABILITY. THE CRITERIA FOR ACCEPTABILITY SHOULD BE MUTUALLY AGREED UPON, MEASUREABLE, AND PUT INTO WRITING. PRIMARILY, ALTHOUGH NOT NECESSARILY, QUALITY IS ASSURED THROUGH SOME FORM OF TESTING. (SEE ALSO QUALITY METRICS)

QUALITY FACTORS
CORRECTNESS, RELIABILITY, EFFICIENCY, INTEGRITY, USABILITY, MAINTAINABILITY, TESTABILITY, FLEXIBILITY, PORTABILITY, REUSABILITY, INTEROPERABILITY, ETC.
SEE ALSO: QUALITY

QUALITY METRICS
MEASURES OF THE CRITERIA OR SUBCRITERIA RELATED TO THE SOFTWARE QUALITY FACTORS. (DAN 283, VOL 1, P 2-2.)

QUEUE
A STORAGE MECHANISM IN A MULTI-USER ENVIRONMENT THAT HOLDS JOBS OR DATA TO BE PROCESSED WITHIN THE OPERATION OF A COMPUTER OR PROGRAM. MOST COMMON ONES ARE TERMED "FIRST IN-FIRST OUT" (FIFO) AND "LAST IN-FIRST OUT" (LIFO). IN SOFTWARE IT IS MORE OFTEN CALLED A "STACK". USED IN FIRST-IN FIRST-OUT (FIFO) LIST ALGORITHMS. (DAN 1153)

RANGE IN MODULE SIZE
THE NUMBER OF SOURCE STATEMENTS IN A MODULE, INCLUDING COMMENTS. (SEL)

RAYLEIGH DISTRIBUTION MODEL
INDEXING TERM. REFERS TO THE MATHEMATICAL METHODOLOGY WHICH IS USED TO CONSTRUCT, OR WHICH IS THE FORM ASSUMED BY, A PARTICULAR MODEL.

READ
THE READING BY PEERS OF THE RECORDINGS OF THE CURRENT PHASE TO LOOK FOR ERRORS, INVENT TEST, ETC. (SEL) (ISO) TO ACQUIRE OR TO INTERPRET DATA FROM A

STORAGE DEVICE, FROM A DATA MEDIUM, OR FROM ANOTHER SOURCE. (ANSI-X3) SEE ALSO CODE READING.

REAL TIME
THIS CLASS INCLUDES SOFTWARE COMPONENTS WHICH ARE DIRECTLY A FUNCTION OF EVENTS OCCURRING AT, OR NEAR, THE CURRENT TIME. TYPICAL COMPONENTS WOULD BE THE ATTITUDE CONTROL MONITORS. SINCE PARTS OF MOST OF THE TELEMETRY PROCESSORS ARE REQUIRED TO PROCESS DATA AS IT IS RECEIVED THEY TOO MAY BE CONSIDERED REAL TIME. (SEL) PERTAINING TO THE PERFORMANCE OF A COMPUTATION DURING THE ACTUAL TIME THAT THE RELATED PHYSICAL PROCESS TRANSPIRES, IN ORDER THAT RESULTS OF THE COMPUTATION CAN BE USED IN GUIDING THE PHYSICAL PROCESS. (ANSI-X3) (3) THE OPERATION OF A COMPUTER OR COMPUTER PROGRAM IN SUCH A WAY THAT COMPUTATIONS ARE SYNCHRONIZED WITH A PHYSICAL PROCESS OR ACTIVITY.(NASA)

REAL TIME PROCESSING
A TYPE OF PROCESSING WHERE RESPONSES ARE REQUIRED TO STIMULI IN A TIME PERIOD (USUALLY SHORT). USUALLY HELD TO BE THE TYPE OF PROCESSING ASSOCIATED WITH THE CONTROL OF A PHYSICAL PROCESS. (ANSI-X3H1)

REAL-TIME EXECUTIVE PROGRAM
THE PORTION OF A DFCAS COMPUTER PROGRAM WHICH CONTROLS THE TIMING OF VARIOUS COMPUTATIONS AND OPERATIONS. (NASA)

REAL-TIME SYSTEMS
A REAL TIME SYSTEM IS A SYSTEM THAT INTERACTS WITH A PHYSICAL ENVIRONMENT TO PERFORM A USEFUL SERVICE FOR THAT ENVIRONMENT OR TO ADEQUATELY CONTROL THAT ENVIRONMENT. ENVIRONMENTAL INTERACTION AND THE NEED TO MAINTAIN ADEQUACY OF CONTROL FREQUENTLY (BUT NOT ALWAYS) TRANSLATE TO STRINGENT RESPONSE-TIME REQUIREMENTS. (DAN 303)

RECORD GENERATORS
A COMPUTER PROGRAM USED TO CONSTRUCT TEST DATA. ESSENTIALLY, THE PROGRAM CONTAINS A LIBRARY OF DATE FORMATS, INCLUDING THE LOCATION, SIZE, CHARACTER (ALPHA OR NUMERIC) AND NORMAL CONTENTS OF EACH FIELD OF EACH RECORD TYPE FROM WHICH IT GENERATES RECORDS REQUIRED FOR TESTING. (DAN 134)

RECOVERY BLOCK STRUCTURED SOFTWARE
A SOFTWARE STRUCTURE INCORPORATING REDUNDANT FAULT-TOLERANT PROVISIONS FOR FLIGHT CRITICAL APPLICATIONS MODULES, NON-REDUNDANT MODULES WITH ERROR DETECTION AND FLAGGING FOR NON-CRITICAL FUNCTIONS, AND A REDUNDANT FAULT-TOLERANT TASK SCHEDULER. (DAN 225) (2) THE RECOVERY BLOCK IS A MEANS OF INDICATING PORTIONS OF A SYSTEM WHOSE OPERATION MUST PASS A DYNAMIC ACCEPTANCE TEST DESIGNED TO DETERMINE PROPER OPERATION. IF THE SYSTEM FAILS THE TEST, AN ALTERNATIVE PROGRAM FOR ACHIEVING THE DESIRED RESULT IS AUTOMATICALLY EXECUTED. (DAN 668)

RECURSION
IN PROGRAMMING, RECURSION REFERS TO THE REPETITIVE SELF-CALLING OF A FUNCTION UPON ITSELF (UNTIL A TERMINATION POINT IS REACHED).

REDUNDANCY
REDUNDANCY IS THE RATIO OF THE QUANTITY OF A PARTICULAR RESOURCE USED IN A SYSTEM TO THE QUANTITY OF THE RESOURCE ACTUALLY NEEDED TO ACCOMPLISH THE

SYSTEMS TASK. REDUNDANCY MAY BE A DESIRABLE CHARACTERISTIC (ERROR DETECTION, ERROR CORRECTION FAULT-TOLERANCE) OR UNDESIRABLE (CAPACITY IN EXCESS OF WHAT MAY EVER BE NEEDED, DUPLICATE DATA FILES FOR PROCESSES WHICH COULD USE THE SAME FILES, ETC.). REDUNDANCY MAY REFER TO DATA, CODE, OR HARDWARE DEVICES.

REGRESSION TESTING

REGRESSION TESTING (RT) IS A METHOD FOR DETECTING ERRORS SPAWNED BY CHANGES OR CORRECTIONS MADE DURING SOFTWARE DEVELOPMENT AND MAINTENANCE. A SET OF TESTS WHICH THE PROGRAM HAS EXECUTED CORRECTLY IS RERUN AFTER EACH SET OF CHANGES IS COMPLETED. IF NO ERRORS OCCUR, CONFIDENCE IS INCREASED THAT SPAWNED ERRORS WERE NOT CREATED IN THAT CHANGE... RT IS AN INVALUABLE AID DURING PROGRAM MAINTENANCE TO PREVENT THE "X STEP FORWARD, Y STEPS BACKWARD" SYNDROME. SPAWNED ERRORS ARE PARTICULARLY ONEROUS FROM A PROGRAM USER POINT OF VIEW, SINCE THEY CONTRIBUTE TO USER DISTRUST ("IT USED TO WORK; WHY DOESN'T IT NOW?). RT IS PRIMARILY USED IN A MAINTENANCE-INTENSIVE ENVIRONMENT. HOWEVER, IT HAS APPLICABILITY TO ANY PROGRAM IN MAINTENANCE, REGARDLESS OF THE QUANTITY OR FREQUENCY OF CHANGE... A SET OF TESTS IS MAINTAINED AND UTILIZED PRIOR TO RELEASE OF EACH NEW SOFTWARE VERSION. IF ERRORS OR DEVIATIONS ARE DETECTED, THEY ARE CORRECTED AND THE REGRESSION TEST IS REPEATED PRIOR TO RELEASE. IF ACCEPTANCE TESTS ARE USED, THEY SHOULD FORM THE BASIS FOR THE REGRESSION TESTS. TESTS SHOULD BE ADDED AS NEW SOFT SPOTS ARE IDENTIFIED DURING MAINTENANCE. BECAUSE OF THE FREQUENCY OF RERUNNING, TESTS SHOULD BE SELF CHECKING WHENEVER POSSIBLE. ALSO SEE - TESTING. (SET)

RELIABILITY DATA

INDEXING TERM. REFERS TO DATA USED TO EVALUATE RELIABILITY, OR TO DRIVE ANY ONE OF THE VARIOUS RELIABILITY MODELS.

RELIABILITY ESTIMATION

RELIABILITY ESTIMATION, IS PERFORMED BY TAKING SOFTWARE RELIABILITY MEASUREMENTS ON AN EXISTING PROGRAM AND MODIFYING THE RESULT TO REPRESENT THE RELIABILITY IN A DIFFERENT OPERATING ENVIRONMENT. A TYPICAL APPLICATION FOR RELIABILITY ESTIMATION IS TO DETERMINE DURING TEST WHETHER AN OPERATIONAL RELIABILITY GOAL CAN BE MET. (2) (FOR COMPUTER PROGRAMS) - THE PROJECTION OF MACROSCOPIC SOFTWARE RELIABILITY DURING THE SOFTWARE DEVELOPMENT PROCESS BASED UPON A MODEL WHICH EMPLOYS PARAMETERS SUCH AS SOFTWARE ERROR DETECTION RATE PER UNIT TIME. EXAMPLE: EMPIRICAL MODEL. (NASA)

RELIABILITY EVALUATION

A COLLECTIVE TERM WHICH ENCOMPASSES ALL OF THE SOFTWARE RELIABILITY METRICS RELATING TO RELIABILITY PREDICTION, RELIABILITY MEASUREMENT, RELIABILITY MODELING, AND RELIABILITY ESTIMATION.

RELIABILITY INDEX

THE PROBABILITY THAT A PROGRAM OR DEVICE WILL PERFORM WITHOUT FAILURE FOR A SPECIFIED PERIOD OF TIME OR AMOUNT OF USAGE. (DAN 1153)

RELIABILITY MEASUREMENT

FOR RELIABILITY MEASUREMENT, THE SOFTWARE IS OPERATED OVER A PERIOD OF TIME, SEGMENTS OF THE OPERATION ARE SCORED AS FAILURE OR SUCCESS, AND FROM THESE SCORES A SINGLE INDICATOR OF MEASURED RELIABILITY IS GENERATED. (2) (FOR COMPUTER PROGRAMS) - THE ASSESSMENT OF SOFTWARE RELIABILITY BASED UPON

SOFTWARE ERRORS DETECTED DURING OPERATIONALLY REPRESENTATIVE EXECUTION OF A PROGRAM. USUALLY DURING DEMONSTRATION TESTING. EXAMPLE: SYSTEM SIMULATION. (NASA)

RELIABILITY MODELS
A SOFTWARE RELIABILITY MODEL USUALLY REFERS TO THE MATHEMATICAL FORM OF THE EQUATIONS WHICH ARE USED IN ESTIMATING THE NUMBER OF REMAINING ERRORS IN A PARTIALLY DEBUGGED SOFTWARE PACKAGE. (DAN 238)

RELIABILITY PREDICTION
RELIABILITY PREDICTION IS A NUMERICAL STATEMENT ABOUT THE RELIABILITY OF A PROGRAM BASED ON SIZE, COMPLEXITY, AND OTHER GENERAL CHARACTERISTICS RATHER THAN ON DATA OBTAINED FROM THE PROGRAM ITSELF. PREDICTION OF RELIABILITY CAN BE MADE EARLY IN THE PROJECT. IT CAN BE USED FOR RESOURCE ALLOCATION TO MODULES AMONG THE TOTAL SOFTWARE AND FOR HARDWARE/SOFTWARE TRADEOFFS AND OTHER MANAGEMENT PURPOSES. (2) (FOR COMPUTER PROGRAMS) - THE PROJECTION OF MICROSCOPIC SOFTWARE RELIABILITY, POSSIBLY IN TERMS OF ERRORS PER INSTRUCTION, BASED UPON QUANTIFIABLE CHARACTERISTICS OR PHENOMENA EXHIBITED BY A PROGRAM. EXAMPLES: STATISTICS,SOFTWARE METRICS, SOFTWARE PHYSICS.(NASA)

RELIABILITY TREND
DEGREE OF CONSTANCY OF THE FAILURE RATE AND/OR FAILURE RATIO OF A SOFTWARE PROJECT OVER TIME. (DAN 226)

RELIABILITY-DIFFERENCES OF OPINION
FEUDS AND DISPUTES (PUBLISHED) BETWEEN MEMBERS OF THE SOFTWARE COMMUNITY WHICH HAVE AN IMPACT UPON CONCERNS OF DACS (GLOSSARY, THESAURUS,ETC) (INDEXING TERM ONLY)

RELIABLE
SEE RELIABLE SOFTWARE

RELIABLE SOFTWARE
SOFTWARE IS RELIABLE IF ITS USE ENABLES A SYSTEM TO PERFORM WITHIN A SPECIFIED ERROR TOLERANCE. THE ABOVE DEFINITION CAN BE INTERPRETED IN TERMS OF TIME OR IN TERMS OF NUMBER OF EXPOSURES TO A UNIT APPLICATION. IN THE FORMER INTERPRETATION, FREQUENCY OF FAILURE IS EQUATED TO THE FRACTION OF THE TIME THE SOFTWARE IS IN A FAILED STATE. IN THE LATTER INSTANCE, FREQUENCY OF FAILURE IS THE FRACTION OF EXPOSURES IN WHICH THE SOFTWARE PREVENTS A UNIT APPLICATION FROM BEING COMPLETED AS EXPECTED. THE TWO INTERPRETATIONS ARE DIRECTLY RELATED WHEN "TIME" REPRESENTS THE OPERATIONAL USAGE TIME, AND THE NUMBER OF EXPOSURES PER UNIT TIME IS SPECIFIED. SOFTWARE IS RELIABLE ONLY IF NO FAULTS EXIST IN A PROGRAM, ROUTINE, OR "MODULE". FROM THE MICROSCOPIC VIEWPOINT FAULTS ARE THE ACTUAL OR POTENTIAL MANIFESTATIONS OF ERRORS MADE BY THE PROGRAM DESIGNER OR CODER. (SET)

RELOCATABLE MACHINE CODE
A (PORTION OF A) COMPUTER PROGRAM, IN MACHINE LEVEL LANGUAGE, EXPRESSED IN SUCH A WAY THAT THE INSTRUCTIONS AND DATA CAN BE STORED IN MEMORY LOCATIONS ASSIGNED DURING LOADING. (NASA)

REPAIRABILITY
REPAIRABILITY IS THE PROBABILITY THAT A FAILED SYSTEM(S) WILL BE RESTORED TO OPERABLE CONDITION WITHIN A SPECIFIED ACTIVE REPAIR TIME WHEN MAINTENANCE IS

DONE UNDER SPECIFIED CONDITIONS. (DAN781)

REPAIRABLE
A SOFTWARE PRODUCT IS REPAIRABLE TO THE EXTENT THAT A CHANGE TO CORRECT A DEFICIENCY CAN BE LOCALIZED, SO AS TO HAVE MINIMAL INFLUENCE ON OTHER PROGRAM MODULES, LOGIC PATHS,OR DOCUMENTATION. REPAIRABILITY IS A SUBCATEGORY OF MAINTAINABILITY, BUT THE IMPLICATION IS THAT A SOFTWARE PRODUCT BECOMES NON-REPAIRABLE WHEN THE EFFECTS OF A PROPOSED CODE FIX ARE NOT UNDERSTOOD WITH SUFFICIENT CONFIDENCE, OWING TO PREVIOUS POOR MAINTENANCE PRACTICES, INCLUDING LACK OF TRACEABILITY. IN OTHER WORDS, A STATE OF NON-REPAIRABILITY IS REACHED WHEN IT CAN BE CONCLUDED THAT IT IS COST EFFECTIVE TO REDESIGN A SIGNIFICANT PORTION OF THE PROGRAM. ALSO SEE MAINTAINABLE (SET)

REPEATABILITY
A PROPERTY REQUIRED OF SOFTWARE TESTS, THAT EACH TIME THEY ARE EXECUTED, THE RESULTS WILL BE THE SAME (DAN 1201)

REPLUGGING
DYNAMIC (WHILE THE SYSTEM IS RUNNING) REPLACEMENT OR RESTRUCTURING OF A MODULE'S IMPLEMENTATION IF THIS REPLACEMENT OR RESTRUCTURING DOES NOT AFFECT THE MODULE'S SPECIFICATION. (DAN 279)

REPORTING
A MAJOR SUB-DIVISION WITHIN CONFIGURATION CONTROL. THE REPORTING AND DOCUMENTING ACTIVITIES NEEDED TO MONITOR THE STATUS OF CONFIGURATION DURING THE LIFE OF A SYSTEM. (DAN LD7)

REPRESENTATION
A DESIGN AND CODING TECHNIQUE THROUGH WHICH COMPOSITE INFORMATION IS STORED OR CONVEYED THROUGH THE ARRANGEMENT, ORGANIZATION OR JUXTAPOSITION OF COMPONENT ELEMENTS. (ABBOTT)

REQUIREMENTS
A SYSTEM SPECIFICATION WRITTEN BY THE USER TO DEFINE A SYSTEM TO A DEVELOPER (A STATEMENT OF WHAT THE USER (PURCHASER) EXPECTS THE SYSTEM TO INCLUDE AMONG ITS CAPABILITIES.) (SEL-MODIFIED)

REQUIREMENTS ANALYSIS
ANALYSIS PERFORMED TO ENSURE THAT THE DEVELOPER'S SOFTWARE REQUIREMENTS ARE COMPLETELY AND CORRECTLY DEFINED. AS PART OF THIS ACTIVITY, ANALYSTS CHECK EACH SOFTWARE REQUIREMENT FOR CONSISTENCY WITH OTHER REQUIREMENTS AND, WHERE POSSIBLE, TRACE SOFTWARE REQUIREMENTS TO THEIR SOURCE IN SYSTEM REQUIREMENTS, INTERFACES, OR USER NEEDS. EACH REQUIREMENT MUST BE DETERMINED CORRECT BY INDEPENDENT DERIVATION, BY COMPARISION TO SIMILAR EXISTING SYSTEMS, OR BY CONSULTING STANDARD REFERENCES. FOR THOSE SOFTWARE REQUIREMENTS WHOSE VALIDITY CANNOT BE DETERMINED A POSTERIORI, OTHER SPECIALIZED TECHNIQUES SUCH AS MODELING, TIMING, AND SIZING ARE USED. ANALYSTS EVALUATE REQUIREMENT TESTABILITY TO ENSURE THAT MEASURABLE ACCEPTANCE CRITERIA ARE IMPLIED BY EACH SOFTWARE REQUIREMENT. FINALLY, THE ENTIRE SET OF SOFTWARE REQUIREMENTS IS EVALUATED FOR COMPLETENESS AND FOR PROPER ALLOCATION OF REQUIREMENTS TO SOFTWARE FUNCTION. SYNONOMOUS WITH REQUIREMENTS VERIFICATION AND ALSO WITH SYSTEM DESIGN REQUIREMENTS. (2) THE PROCESS OF STUDYING THE CUSTOMER'S PROBLEM FROM BOTH THAT OF THE DEVELOPER

AND THE USER IN ORDER TO ARRIVE AT A FUNCTIONAL DEFINITION OF SYSTEM REQUIREMENTS. INCLUDES ALL ACTIVITIES RELATED TO ANALYZING AND DEVELOPING A CLEAR, UNEQUIVOCAL, AND MUTUALLY-AGREED-UPON SET OF FUNCTIONAL SPECIFICATIONS FOR A PROJECT. (DAN LD7)

REQUIREMENTS AND DESIGN SPECIFICATIONS
PRECISE SPECIFICATIONS OF THE OPERATIONAL CHARACTERISTICS OF THE FINAL PRODUCT. (DAN LD7)

REQUIREMENTS ENGINEERING
THE DISCIPLINE OF APPLYING ENGINEERING, ESPECIALLY, SOFTWARE ENGINEERING, TECHNIQUES TO BOTH THE DEVELOPMENT AND STATEMENT OF REQUIREMENTS. THE OBJECT IS TO BRING VISIBILITY TO THOSE TYPES OF DEFICIENCES IN THE REQUIREMENTS WHICH RECUR REGULARLY ACROSS A BROAD RANGE OF SOFTWARE DEVELOPMENT PROJECTS. ONCE VISIBILITY HAS BEEN ACHIEVED, THESE DEFICIENCES CAN BE ELIMININATED AND THUS, THOSE TYPES OF ERRORS WHICH ARISE FROM INCOMPLETE OR POORLY DEFINED REQUIREMENTS CAN BE PREVENTED.

REQUIREMENTS ENGINEERING METHODOLOGIES
INDEXING TERM. REFERS TO DEVELOPMENTAL METHODOLOGIES WHICH TREAT THE GENERATION OF SOFTWARE REQUIREMENTS AS AN ENGINEERING DISCIPLINE.

REQUIREMENTS LANGUAGE
A COMPUTER PROGRAM USED TO PROVIDE A SUCCINCT AND UNAMBIGUOUS SPECIFICATION OF THE SYSTEM, THEN COMPUTER REQUIREMENTS. IT MORE PRECISELY ALLOWS REQUIREMENTS TO BE COMMUNICATED AND TRANSLATED IN A HIERARCHICAL MANNER. (DAN 134)

REQUIREMENTS PROBLEM CATEGORIES
THE CLASSIFICATION SYSTEM FOR SOFTWARE REQUIREMENTS PROBLEMS.

REQUIREMENTS PROBLEMS
PROBLEMS IN THE SOFTWARE SYSTEM CAUSED BY DEFICIENCIES IN SOFTWARE REQUIREMENTS.

REQUIREMENTS SPECIFICATION
TRANSLATION OF AN OPERATIONAL (OR APPLICATION) REQUIREMENT INTO A STATEMENT OF THE FUNCTIONS TO BE PERFORMED. (DAN 773)

REQUIREMENTS SPECIFICATION LANGUAGE
A LANGUAGE USED TO SPECIFY A SOFTWARE SYSTEM WHICH IS SUFFICIENTLY FORMAL IN THE MATHEMATICAL SENSE, THAT CONCLUSIONS CONCERNING CONSISTENCY AND COMPLETENESS MAY BE DRAWN FROM THE SYSTEM'S SPECIFICATIONS EXPRESSED IN SUCH LANGUAGES. (DAN 874-MOD)

REQUIREMENTS SPECIFICATION SUPPORT TOOL
ANY TOOL WHICH PROVIDES A MEANS FOR EVALUATING COMPLETENESS AND CONSISTENCY OF THE REQUIREMENTS SPECIFICATION TO THE DESIGNER'S SATISFACTION AND FOR DEMONSTRATING BEHAVIOR TO THE CUSTOMER. THE TOOL SHOULD ALSO ASSIST THE DESIGNER IN THE DOCUMENTATION, MANIPULATION, MODIFICATION, AND CATALOGUING OF THE DESIGN THROUGHOUT THE ITERATIONS UNTIL SIGN-CFF. (DAN 874)

REQUIREMENTS TESTING
EXECUTION OF A PROGRAM UNDER CONTROLLED CONDITIONS TO DEMONSTRATE THAT ALL STATED OR IMPLIED REQUIREMENTS AND PERFORMANCE CRITERIA HAVE BEEN MET. (DAN 1153)

REQUIREMENTS VERIFICATION
REQUIREMENTS VERIFICATION (REQVER) IS THE PROCESS OF DETERMINING WHETHER OR NOT THE COMPUTER PROGRAM REQUIREMENTS REFLECT THE COMPUTER-APPLICABLE PORTION OF THE SYSTEM SPECIFICATION. ITS PRIMARY PURPOSE IS TO IDENTIFY AMBIGUOUS, ILL-DEFINED, AND INADEQUATE COMPUTER REQUIREMENTS EARLY IN THE ACQUISITION CYCLE. REQVER SEEKS TO DETERMINE IF THE CONCEPTUAL DESIGN WILL WORK. ITS INTENT IS TO VERIFY THAT EACH REQUIREMENT STATED IN SYSTEM SPECIFICATION IS CLEARLY TRANSLATED INTO SUBSYSTEM REQUIREMENTS THAT CAN BE MECHANIZED. SIMULATIONS, DOCUMENT RESEARCH, AND ANALYSIS ARE THE TECHNIQUES PRESENTLY ASSOCIATED WITH REQVER. (SET)

RESERVATIONS AND DISPATCHING APPLICATIONS
SOFTWARE APPLICATION TO SYSTEMS SUCH AS AIRLINE RESERVATIONS, CAR RENTAL RESERVATIONS, FIRE DEPT'S, POLICE DEPT'S ETC.

RESILIENCY
RESILIENCY REFERS TO SHARED SYSTEMS. A RESILIENT SERVICE: 1) IS ABLE TO DETECT AND RECOVER FROM A GIVEN MAXIMUM NUMBER OF ERRORS, 2) IS RELIABLE TO A SUFFICIENTLY HIGH DEGREE THAT A USER OF THE RESILIENT SERVICE CAN IGNORE THE POSSIBILITY OF SERVICE FAILURE. 3) IF THE SERVICE PROVIDES PERFECT DETECTION AND RECOVERY FROM N ERRORS, THE (N+1)ST ERROR IS NOT CATASTROPHIC. A "BEST EFFORT" IS MADE TO CONTINUE SERVICE, AND 4) THE ABUSE OF THE SERVICE BY A SINGLE USER SHOULD HAVE NEGLIGIBLE EFFECT ON OTHER USERS OF THE SERVICE. (SOURCE UNKNOWN)

RESOURCE
A SOURCE OF SUPPLY OR SUPPORT. (ANSI-X3H1) (2) ANY COMMODITY OR AVAILABLE MEANS THAT MAY BE ALLOCATED TOWARD THE ACCOMPLISHMENT OF A TASK. IN CONCURRENT PROCESSES, SHARED RESOURCES ARE CHARACTERIZED AS "DEVOTED" (ALLOCATED FOR MUTUALLY EXCLUSIVE USE) OR "MUTUAL" (CAN BE ENGAGED IN SIMULTANEOUS OPERATIONS UNDER STATED LIMITATIONS). RESOURCES PRODUCED BY ONE PROCESS AND CONSUMED BY ANOTHER ARE SAID TO BE "TEMPORARY RESOURCES"; OTHER RESOURCES ARE "PERMANENT". (DAN 1153) SEE ALSO: COMPUTER RESOURCES

RESOURCE ALLOCATION
THE AMOUNT OF RESOURCES (TIME, MONEY, PERSONNEL) EXPENDED ON PARTICULAR PORTIONS OF THE SOFTWARE (OR SYSTEM) DEVELOPMENT.

RESPONSE
A MESSAGE IN ANSWER TO OR AS A REACTION TO A PREVIOUS ACTION. (ANSI-X3H1)

RESTRUCTURING ALGORITHMS
AN ALGORITHM WHICH UTILIZES A BLOCK REFERENCE STRING TO PRODUCE A RESTRUCTURING GRAPH WHOSE NODES CORRESPOND TO THE BLOCKS AND WHOSE EDGES HAVE LABELS SOMEHOW REPRESENTING THE DESIRABILITY OF STORING THE TWO BLOCKS THEY CONNECT IN ADJACENT AREAS OF THE VIRTUAL ADDRESS SPACE (DAN 435)

RESYNCHRONIZATION
RESYNCHRONIZATION IS THE PROCESS OF HALTING TWO COMPUTERS AND SIMULTANEOUSLY

RESTARTING BOTH, TO ENSURE THAT BOTH ARE PERFORMING THE IDENTICAL OPERATIONS AT THE SAME TIME. RESYNCHRONIZATION IS A FREQUENTLY USED TECHNIQUE FOR IMPLEMENTING FAULT-TOLERANT SOFTWARE. (DAN 761)

RETEST
VALIDATION OF MAINTENANCE MODIFICATIONS. (FISHER,G)

RETURN
A RETURN IS AN INSTRUCTION THAT RELEASES CONTROL TO THE CALLING ROUTINE OR TO THE EXECUTIVE ROUTINE. (SET)

REUSABILITY
REUSABILITY REFERS TO THE EXTENT TO WHICH A PROGRAM CAN BE USED IN OTHER APPLICATIONS - RELATED TO THE PACKAGING AND SCOPE OF THE FUNCTIONS THAT A PROGRAM PERFORMS. (DAN 512)

REUSABLE CODE
SOURCE CODE ORIGINALLY DEVELOPED FOR ANOTHER APPLICATION THAT CAN BE USED FOR A DIFFERENT APPLICATION.

REVIEW
FORMAL MEETING OF SEVERAL INDIVIDUALS FOR THE PURPOSE OF EVALUATING DESIGN (MANAGEMENT REVIEW). ALSO INCLUDES "HOME WORK" IN PREPARING FOR THE REVIEW. ALL OF THOSE ATTENDING A REVIEW SHOULD HAVE COMPLETED DISCUSSED IN THEIR OWN COMPONENT SUMMARY REPORT FOR THAT MEET. (SEE ALSO: DESIGN REVIEW.

ROBOTICS
UTILIZATION OF A COMPUTER TO CONTROL MECHANICAL MANIPULATION.

ROBUSTNESS
CODE POSSESSES THE CHARACTERISTIC ROBUSTNESS TO THE EXTENT THAT IT CAN CONTINUE TO PERFORM DESPITE SOME VIOLATION OF THE ASSUMPTIONS IN ITS SPECIFICATION. THIS IMPLIES, FOR EXAMPLE, THAT THE PROGRAM WILL TRIVIALLY HANDLE INPUTS OUT OF RANGE, OR IN A TIMELY FASHION TO THEIR TRUE DESIRED, WITHOUT DEGRADING ITS PERFORMANCE OR FUNCTIONS NOT DEPENDENT ON THE NON-STANDARD INPUTS. (DAN 239)

ROLLBACK
A PROGRAMMED RETURN TO A PRIOR CHECKPOINT. (FAUST-83) (SEE ALSO USE OF (DAN 761 FOR DEFINITION WITH/RESP TO FAULT-TOLERANT SOFTWARE).

ROUTINE
A PROGRAM OR PROGRAM MODULE THAT MAY HAVE SOME GENERAL OR FREQUENT USE. IN A PROGRAM MODULE, THEN A ROUTINE ALWAYS RETURNS TO THE POINT OF INVOCATION AFTER EXECUTION (I.E., A PROPER ROUTINE) OR ABNORMALLY TERMINATE. (DAN 1153) (2) SMALLEST GROUP OF COMPILABLE CODE. (DAN 21)

ROUTING DATA
ROUTING DATA, ALSO CALLED "TRACKING DATA" OR "TRAFFIC DATA" PROVIDES AN ACCOUNT OF THE RECIPIENTS OF DATA STORED IN A RECORD ASSOCIATED WITH SOME INDIVIDUAL. (DAN 617)

SADM
(SYSTEMATIC ACTIVITY MODELING METHOD) A DESIGN REPRESENTATION SCHEME WHOSE

OBJECTIVE IS TO MODEL A SYSTEM USING HIERARCHICAL DECOMPOSITION AND DATA FLOW. THE SCHEME IS COMPOSED OF THREE ELEMENTS; A HIERARCHICAL TREE STRUCTURE WHICH DESCRIBES HOW A PARTICULAR DIAGRAM FITS INTO THE SYSTEM, AN ACTIVITY DATAFLOW DIAGRAM WHICH DESCRIBES THE ACTIVITY-DATA FLOW RELATIONSHIPS OF A SYSTEM, AND A CONDITION CHART WHICH DOCUMENTS THE FUNCTIONAL BEHAVIOR OF A DIAGRAM. THE AUTOMATION PACKAGE FOR THE SAMM DESIGN REPRESENTATION SCHEME IS CALLED SAMDF.

SARA
(SYSTEMS ARCHITECT'S APPRENTICE) A SET OF INTERACTIVE DESIGN MODELING TOOLS WHICH SUPPORT A STRUCTURED MULTILEVEL DESIGN PROCEDURE FOR SOFTWARE OR HARDWARE DEVELOPMENT. (DAN 272)

SCENARIO
AN AUTOMATED TEST CONTROL PACKAGE CONSISTING OF TEST EXECUTION CONTROL WORDS, TEST DATA, AND EVENT STIMULI USED TO ACTIVATE AND TEST A TARGET PROGRAM. (DAN 1201)

SCHEDULE ESTIMATION
THE PROCESS OF DETERMINING THE AMOUNT OF TIME NEEDED TO COMPLETE A GIVEN PROJECT. THE PROCESS USUALLY ALSO INCLUDES FORECASTING THE TIME NEEDED TO REACH VARIOUS MILESTONES IN THE PROJECT. THERE ARE TWO APPROACHES; THE MACRO APPROACH WHICH GENERATES AN EXPECTED CURVE OF LIFE-CYCLE MANPOWER AGAINST TIME, AND THE MICRO APPROACH WHICH STARTS WITH FIXING THE SIZE, START DATE, AND DURATION OF EACH DISTINGUISHABLE ACTIVITY, MAKES ADJUSTMENTS FOR CALIBER OF PERSONNEL, COMPLEXITY, AND OTHER FACTORS, ARRANGES THE FACTORS IN A NETWORK, AND SELECTS AS THE SCHEDULE TIME, THE LENGTH OF THE LONGEST PATH IN THE NETWORK. (DAN 224)

SCHEDULE MANAGEMENT
A METHOD OF DETERMINING WHAT WORK MUST BE DONE TO PRODUCE EACH ELEMENT OF A SYSTEM. (DAN 167)

SCHEDULING
IN OPERATING SYSTEMS, SCHEDULING REFERS TO THE ALLOCATION OF JOBS FROM A QUEUE TO RUN IN AN EFFICIENT OR PRIORITIZED MANNER.

SCHICK-WOLVERTON MODEL
A SOFTWARE RELIABILITY MODEL DEVELOPED BY DRS. H. WOLVERTON AND G. SCHICK OF TRW. THE BASIC ASSUMPTIONS FOR THE SCHICK-WOLVERTON MODEL ARE: 1) THE AMOUNT OF DEBUGGING TIME BETWEEN ERROR OCCURRENCES HAS A RAYLEIGH DISTRIBUTION. 2) THE ERROR RATE IS PROPORTIONAL TO THE NUMBER OF ERRORS REMAINING AND TO THE TIME SPENT IN DEBUGGING. 3) EACH ERROR DISCOVERED IS IMMEDIATELY REMOVED, THUS REDUCING THE NUMBER OF ERRORS BY ONE. THE MODIFIED SCHICK-WOLVERTON MODEL IS SIMILAR TO THE ABOVE WITH THE EXCEPTION THAT ASSUMPTION 2 IS REPLACED BY THE FOLLOWING ASSUMPTION; 2) THE ERROR DISCOVERY RATE IS CONSTANT DURING A TIME INTERVAL AND IS PROPORTIONAL TO THE NUMBER OF ERRORS REMAINING FOLLOWING THE (I-1)ST TIME INTERVAL AND THE TOTAL TIME PREVIOUSLY SPENT IN TESTING, INCLUDING AN "AVERAGED" ERROR SEARCH TIME DURING THE CURRENT DEBUG INTERVAL I. (DAN 402)

SCIENTIFIC
A SOFTWARE COMPONENT MAY BE IN THIS CATEGORY IF IT IS RELATED TO SOME MATHEMATICAL ALGORITHM, ENGINEERING PROBLEM, LAW OF PHYSICS, OR CELESTIAL

MECHANICS PROBLEM. MOST ALL OF THE FULL SYSTEMS DEVELOPED WILL FALL INTO THIS CATEGORY, WHILE THE VARIOUS PIECES OR MODULES MAY FALL INTO SOME OF THE OTHER CLASSES. [SPECIFIC TO NASA-GODDARD] (SEL)

SCOPE
THE RANGE WITHIN WHICH AN IDENTIFIED UNIT DISPLAYS ITSELF. SCOPE OF ACTIVITY REFERS TO THE BOUNDARIES WITHIN WHICH A DATA STRUCTURE OR PROGRAM ELEMENT REMAINS AN INTEGRAL UNIT. SCOPE OF CONTROL REFERS TO THE SUBMODULES IN A PROGRAM THAT POTENTIALLY MAY EXECUTE IF CONTROL IS GIVEN TO A CITED MODULE. SCOPE OF ERROR DENOTES THE SET OF SUBMODULES THAT ARE POTENTIALLY AFFECTED BY THE DETECTION OF A FAULT IN A CITED MODULE. (DAN 1153)

SCORING PROGRAM
A COMPUTER PROGRAM THAT PERFORMS THE SAME CALCULATIONS AS THE TARGET SYSTEM. THE PROGRAM INCLUDES A COMPARE CAPABILITY SO THAT RESULTS COMPUTED BY THE TARGET SYSTEM CAN BE AUTOMATICALLY COMPARED AND DISCREPANCIES FLAGGED. (DAN 134)

SECRETARIAT
A CENTRALIZED FACILITY CONSISTING OF PROCESSING AIDS, LIBRARY MATERIALS, AND PRODUCTION SERVICES AVAILABLE TO DEVELOPMENT PROJECTS, FOR THE PURPOSE OF RAISING PRODUCTIVITY, ENFORCING STANDARDS, AND MONITORING PROGRESS. (DAN 1153)

SEMANTICS
THE RELATION BETWEEN SYMBOLS AND THEIR MEANINGS. (ANSI-X3H1)

SEMAPHORE
A SHARED DATA STRUCTURE USED BY CONCURRENT PROCESSES TO EFFECT SYNCHRONIZATION, CONSISTING OF AN ARBITRATED VARIABLE THAT CONTAINS THE NET NUMBER OF "MESSAGES" SENT, NOT YET RECEIVED, AND A QUEUE THAT CONTAINS A LIST (IF NOT EMPTY) OF PROCESSES CURRENTLY WAITING FOR A "MESSAGE". (DAN 1153)

SEQUENTIAL TESTING PROCEDURE
A PROCEDURE FOR SEARCHING A DECISION TABLE CONDITION ENTRY TO DETERMINE WHICH RULE APPLIES TO A GIVEN ARRAY OF ANSWERS TO THE CONDITION STUB. THUS, AN ALGORITHM FOR PROCESSING THE UPPER HALF OF A DECISION TABLE TO DETERMINE WHICH RULE IS TO BE ACTIVATED. (DAN 1153)

SESSION
A PERIOD OF A GIVEN USER'S ACCESS TO, AND USE OF, A [illegible] SYSTEM. (ANSI-X3H1)

SIDE-EFFECT
A SECONDARY EFFECT DUE TO CONNECTIVITY AMONG MODULES WHICH, THEREFORE, PROPAGATES IN THE SAME MODE AS PROGRAM CONNECTIONS: CONTROL, DATA, SERVICES. CONTROL SIDE EFFECTS ARISE IN NON-PROPER PROGRAMMING, DATA CONNECTION SIDE EFFECTS ARISE IN USE OF COMMON, EXTERNAL COUPLING, CONTENT COUPLING, AND NOT UTILIZING THE NORMAL PARAMETER PASSING MECHANISM; AND SERVICE SIDE EFFECTS ARISE WHEN THE ROLE OR ACTION OF A PROCEDURE VARIES WITH ITS APPLICATION (AS A RESULT OF GLOBAL VARIABLES MODIFIED, LOCAL DATA MODIFIED AND RETAINED, OR CHANGES IN HARDWARE STATUS). (DAN 1153)

SIGN OFF
TO INDICATE THE END OF A PERIOD OF IDENTIFICATION TO THE SYSTEM. (ANSI-X3H1) SEE ALSO SIGN ON.

SIGN ON
TO IDENTIFY ONESELF TO A SYSTEM, AND TO BE ACCEPTED BY THAT SYSTEM. (ANSI-X3H1)

SIMULATION
THE PRACTICE OF EMULATING THE PRESENCE AND ACTIONS OF THE INTERFACE NORMALLY PRESENTED BY A SOFTWARE MODULE OR HARDWARE DEVICE. (DAN 1201)

SIMULATION MODULE/PROGRAM
A COMPUTER PROGRAM THAT PROVIDES THE TARGET SYSTEM WITH INPUTS OR RESPONSES THAT RESEMBLE THOSE THAT HAVE BEEN PROVIDED BY THE PROCESS FOR THE DEVICE BEING SIMULATED. (2) A PROGRAM WHOSE PURPOSE IS TO IMITATE THE INTERFACE OF OTHER SOFTWARE MODULES OR HARDWARE DEVICES. (DAN 1201)

SINGLE ENTRY SINGLE EXIT CONTROL STATEMENT
ANY CONTROL STATEMENT WHICH DEFINES A CONTROL STRUCTURE. (ABBOTT)

SOFTWARE
SOFTWARE IS COMPUTER PROGRAM CODE AND ITS ASSOCIATED DATA, DOCUMENTATION, AND OPERATIONAL PROCEDURES. (SET)

SOFTWARE ANALYSIS
THE PRELIMINARY DESIGN FOR A SOFTWARE SYSTEM (DAN 258)

SOFTWARE DATA BASE
A COMMON DATA BASE INTERNAL TO AN OPERATIONAL SOFTWARE SYSTEM. (DAN 300) (2) A DATABASE DEVOTED TO DOCUMENTS AND INFORMATION ABOUT SOFTWARE.

SOFTWARE DESIGN DEFINITION
(SDD) A DOCUMENT CHIEFLY USED TO DISPLAY THE RESULTS OF THE ARCHITECTURAL DESIGN STUDY AND IMPLICATIONS OF COST, SCHEDULE, AND WORK-BREAKDOWN STRUCTURES TO MANAGEMENT OR CUSTOMERS. SOME HIGH-LEVEL TECHNICAL MATERIAL, SUCH AS THAT NEEDED TO ASSESS PROBLEM AREAS AND OTHER CONCERNS OR TO SHOW HOW REQUIREMENTS WILL BE MET, IS INCLUDED. (DAN 1153)

SOFTWARE DESIGN METHODOLOGY
A SYSTEMATIZED PROCEDURE FOR CARRYING OUT THE OVERALL SOFTWARE DEVELOPMENT PROCESS THROUGH THE EFFECTIVE USE OF AN INTEGRATED COLLECTION OF TOOLS AND TECHNIQUES. (NASA)

SOFTWARE DEVELOPMENT CYCLE
SOFTWARE DEVELOPMENT CYCLE 1) REQUIREMENTS SPECIFICATION. TRANSLATION OF AN OPERATIONAL (OR APPLICATION) REQUIREMENT INTO A STATEMENT OF THE FUNCTIONS TO BE PERFORMED. 2) SYSTEM DESIGN. TRANSLATION OF THE REQUIREMENTS INTO A DESCRIPTION OF ALL THE COMPONENTS NECESSARY TO IMPLEMENT THE SYSTEM. 3) PROGRAMMING. PRODUCTION OF THE CODE WHICH WILL CONTROL THE SYSTEM AND PERFORM ALL REQUIRED LOGIC AND COMPUTATION. 4) CHECKOUT. VERIFICATION THAT THE CODED AND OTHER COMPONENTS OF THE SYSTEM SATISFY THE ORIGINAL REQUIREMENTS. (DAN 773) (2) THE SOFTWARE DEVELOPMENT CYCLE CONSIST OF FOUR PHASES: DEFINITION, DESIGN, IMPLEMENTATION AND EVALUATION. (DAN 141)

SOFTWARE DEVELOPMENT LIBRARY
(SDL) A PROJECT INTERNAL FACILITY FOR INTERFACE AND MANAGEMENT VISIBILITY AND FOR INTERFACE AND MANAGEMENT VISIBILITY AND FOR SOFTWARE PRODUCTION MANAGEMENT AND CONTROL. (DAN 1153) SEE ALSO: PROGRAM SUPPORT LIBRARY, PROGRAMMING SUPPORT LIBRARY

SOFTWARE DEVELOPMENT PROCESS
THE SOFTWARE DEVELOPMENT PROCESS IS THE FORMAL PROCESS BY WHICH THE SOFTWARE REQUIREMENTS ARE TRANSFORMED INTO DESIGN SPECIFICATIONS, IMPLEMENTED INTO CODE, TESTED, DOCUMENTED, AND FINALLY PLACED IN OPERATIONAL STATUS. SEE ALSO: SOFTWARE LIFE CYCLE

SOFTWARE DOCUMENTATION
SOFTWARE DOCUMENTATION IS TECHNICAL DATA, INCLUDING COMPUTER LISTINGS AND PRINTOUTS, IN HUMAN-READABLE FORM WHICH (1) DOCUMENTS THE DESIGN OR DETAILS OF THE SOFTWARE, (2) EXPLAINS THE CAPABILITIES OF THE SOFTWARE, OR (3) PROVIDES OPERATING INSTRUCTIONS FOR USING THE SOFTWARE TO OBTAIN DESIRED RESULTS FROM COMPUTER EQUIPMENT. (SET)

SOFTWARE ENGINEERING
SOFTWARE ENGINEERING COMBINES THE USE OF MATHEMATICS TO ANALYZE AND CERTIFY ALGORITHMS, ENGINEERING TO ESTIMATE COSTS AND DEFINE TRADEOFFS, AND MANAGEMENT SCIENCE TO DEFINE REQUIREMENTS, ASSESS RISKS, OVERSEE PERSONNEL, AND MONITOR PROGRESS IN THE DESIGN, DEVELOPMENT AND USE OF SOFTWARE. SOFTWARE ENGINEERING TECHNIQUES ARE DIRECTED TO REDUCING HIGH SOFTWARE COST AND COMPLEXITY WHILE INCREASING RELIABILITY AND MODIFIABILITY. (2) SOFTWARE ENGINEERING IS THAT BRANCH OF SCIENCE AND TECHNOLOGY WHICH DEALS WITH THE DESIGN, DEVELOPMENT, AND USE OF SOFTWARE. SOFTWARE ENGINEERING IS A DISCIPLINE DIRECTED AT THE PRODUCTION OF COMPUTER PROGRAMS THAT ARE CORRECT, EFFICIENT, FLEXIBLE, MAINTAINABLE, AND UNDERSTANDABLE IN REASONABLE TIME SPANS AT ACCEPTABLE COSTS. (SET) (3) THE PRACTICAL AND METHODICAL APPLICATION OF SCIENCE AND TECHNOLOGY IN THE DESIGN, DEVELOPMENT, EVALUATION, AND MAINTENANCE OF COMPUTER SOFTWARE OVER ITS LIFE CYCLE. (NASA)

SOFTWARE ENGINEERING BLUEPRINTS
A PROGRAM DESIGN REPRESENTATION WHICH PROVIDES AN "EASILY READABLE" AND COMPREHENSIVE "LAYOUT" OF THE FUNCTIONAL ARCHITECTURE OF A SOFTWARE PRODUCT. (DAN 271)

SOFTWARE ENGINEERING FACILITY
AN ORGANIZATION OR COMPANY WHOSE PRODUCT IS SOFTWARE DEVELOPMENT AND PRODUCTION. (DAN 253)

SOFTWARE ENGINEERING MANAGEMENT
THE JUDICIOUS USE OF MEANS TO EFFECT AND ADMINISTER THE ADVANCEMENT OR USAGE OF INFORMATION SYSTEMS TECHNOLOGY. SOFTWARE ENGINEERING MANAGEMENT RECOGNIZES NEEDS, SETS GOALS, PLANS MODES OF ACCOMPLISHMENT, DEVISES MEANS FOR RESOURCE ALLOCATION, AND DIRECTS THE APPROACH TAKEN IN FUTURE INFORMATION SYSTEMS APPLICATIONS AND IN THE SOLUTION OF PROBLEMS ASSOCIATED WITH THESE APPLICATIONS. (DAN 1153)

SOFTWARE ENGINEERING PROJECT MANAGEMENT
SOFTWARE ENGINEERING PROJECT MANAGEMENT IS MANAGEMENT WHICH PROVIDES THE NECESSARY PLANNING, ORGANIZATION, STAFFING, DIRECTION AND CONTROL FOR THE

ORDERLY DEVELOPMENT OF A SOFTWARE PROJECT. (DAN 230)

SOFTWARE ENGINEERING STANDARDS
A SET OF ENFORCEABLE GUIDELINES AND/OR CONSTRAINTS USED TO INSURE CONSISTENCY DURING THE DEVELOPMENT PERIOD, THEY MAY APPLY TO SUCH AREAS AS NOMENCLATURE, NAMING, LABELING, CODING, COMPUTER USAGE, DOCUMENTATION, PROCEDURES, & TESTING. (DAN 300)

SOFTWARE ENGINEERING TOOLS AND TECHNIQUES
INDEXING TERM. REFERS TO APPLICATION OF SOFTWARE TOOLS AND DEVELOPMENTAL TECHNIQUES IN A SOFTWARE ENGINEERING PROJECT AS AN INTEGRAL PART OF THE DEVELOPMENTAL PROCESS.

SOFTWARE ERRORS
ANY DISCREPANCY BETWEEN A COMPUTED, OBSERVED OR MEASURED QUANTITY AND ITS TRUE, SPECIFIED, OR THEORETICALLY CORRECT VALUE. ERRORS ARE INTRODUCED INTO SOFTWARE BY HUMAN MISTAKES THAT IS: DEFICIENCIES OR MISINTERPRETATIONS OF DESIGN CRITERIA, LOGICAL MISTAKES, SYNTATICAL MISTAKES MADE IN TRANSCRIBING PROGRAM STATEMENTS INTO THE INPUT DATA. (DAN LD4)

SOFTWARE EXPERIENCE DATA
DATA RELATING TO THE DEVELOPMENT OR USE OF SOFTWARE WHICH COULD BE USEFUL IN DEVELOPINTG MODELS, RELIABILITY PREDICTIONS OR OTHER QUANTITATIVE DESCRIPTIONS OF SOFTWARE.

SOFTWARE FACTORY
A COLLECTED SET OF TOOLS, METHODOLOGIES, AND A COMMON DATA BASE THAT PROVIDE A PROCEDURAL APPROACH TO THE SUCCESSFUL COMPLETION OF SOFTWARE PROJECTS. (DAN LD7)

SOFTWARE FAMILIES
A FLEXIBLE FAMILY OF SOFTWARE PACKAGES THAT CAN BE TARGETED TO VARIOUS MACHINES AND THAT CAN BE REHOSTED. (DAN 381)

SOFTWARE LIBRARY MANAGEMENT SYSTEM
A CENTRAL COMPONENT OF ANY SOFTWARE DEVELOPMENT SYSTEM WHICH STORES SUCH THINGS AS THE REQUIREMENTS FOR THE COMPUTER SYSTEM, SOURCE MODULES, OBJECT MODULES, DOCUMENTATION, INPUT DATA, OUTPUT DATA AND CONFIGURATIONS AND THEIR ATTENDANT CHANGE NOTICES. (DAN 366)

SOFTWARE LIFE CYCLE
THE SOFTWARE LIFE CYCLE IS THAT PERIOD OF TIME IN WHICH THE SOFTWARE IS CONCEIVED, DEVELOPED, AND USED. (2) THE LIFE CYCLE IS NORMALLY DIVIDED INTO THE SIX PHASES OF CONCEPTION, REQUIREMENTS DEFINITION, DESIGN, IMPLEMENTATION, TEST, AND OPERATIONAL PHASES. THE CONCEPTUAL PHASE ENCOMPASSES PROBLEM STATEMENT DEFINITION, PRELIMINARY SYSTEMS ANALYSIS, AND THE IDENTIFICATION OF ALTERNATIVE SOLUTION CATEGORIES. THE REQUIREMENTS DEFINITION PHASE CONSISTS OF PRODUCING A STATEMENT OF PROJECT OBJECTIVES, SYSTEM FUNCTIONAL SPECIFICATIONS, AND DESIGN CONSTRAINTS. DURING THE DESIGN PHASE THE SOFTWARE COMPONENT DEFINITIONS, INTERFACE, AND DATA DEFINITIONS ARE GENERATED AND VERIFIED AGAINST THE REQUIREMENTS. THE IMPLEMENTATION PHASE CONSISTS OF THE ACTUAL PROGRAM CODE GENERATION, UNIT TESTING OF THE PROGRAMS, AND DOCUMENTING THE SYSTEM. DURING THE TEST PHASE, SYSTEM INTEGRATION OF THE SOFTWARE COMPONENTS AND SYSTEM ACCEPTANCE TESTS ARE

PERFORMED AGAINST THE REQUIREMENTS. THE OPERATIONAL PHASE INVOLVES THE USE AND MAINTENANCE OF THE SYSTEM. THIS INCLUDES THE DETECTION AND CORRECTION OF ERRORS AND THE INCORPORATION OF MODIFICATIONS TO ADD CAPABILITIES AND/OR IMPROVE PERFORMANCE... ALSO SEE - COMPUTER PROGRAM CERTIFICATION, COMPUTER PROGRAM VALIDATION, COMPUTER VERIFICATION, MAINTAINABLE TESTING, PROGRAMMING, SOFTWARE DEVELOPMENT PROCESS. (SET)

SOFTWARE MONITOR

A COMPUTER PROGRAM THAT PROVIDES DETAILED STATISTICS ABOUT SYSTEM PERFORMANCE. BECAUSE SOFTWARE MONITORS RESIDE IN MEMORY, THEY HAVE ACCESS TO ALL THE TABLES THE SYSTEM MAINTAINS. THEREFORE, THEY CAN EASILY EXAMINE SUCH THINGS AS CORE USAGE, QUEUE LENGTHS, INDIVIDUAL PROGRAM OPERATION, AND SO ON TO HELP MEASURE PERFORMANCE. (DAN 134) COMPARE WITH SNAP GENERATOR, SNAPS, TRACE, TRACER PROGRAM.

SOFTWARE PHYSICS

FUNDAMENTAL DEFINITION - ONE UNIT OF SOFTWARE WORK IS PERFORMED ON A STORAGE MEDIUM WHEN ONE BYTE OF THAT MEDIUM IS ALTERED. THE ENERGY OF A PROCESSOR IS THE CAPACITY OF THAT PROCESSOR TO PERFORM COMPUTATION WORK. (TIES TOGETHER 3 CONCEPTS: SOFTWARE WORK, POWER, STORAGE REALIZATION) BY MEASURING, THE RATE AT WHICH ENERGY IS CONVERTED TO WORK, WE DEFINE THE POWER OF A PROCESSOR. A STORAGE REALIZATION IS THE PATTERN OF BITS OF MEMORY WHERE THEY HOLD THE INSTRUCTIONS AND DATA OF A GIVEN PROGRAM. (DAN 231)

SOFTWARE PRODUCT

THE SOFTWARE COMPONENT OF A DELIVERABLE (TO A CUSTOMER) PRODUCT.

SOFTWARE RELIABILITY

SOFTWARE RELIABILITY IS THE PROBABILITY THAT A GIVEN SOFTWARE PROGRAM WILL OPERATE WITHOUT FAILURE FOR A SPECIFIED TIME IN A SPECIFIED ENVIRONMENT. (2) SOFTWARE RELIABILITY IS DEFINED AS THE PROBABILITY THAT A GIVEN SOFTWARE PROGRAM OPERATES FOR SOMETIME PERIOD, WITHOUT AN EXTERNAL SOFTWARE ERROR, ON THE MACHINE FOR WHICH IT WAS DESIGNED GIVEN THAT IT IS USED WITHIN DESIGN LIMITS. (DAN 31). (3) A. SYSTEM, USER, OR MACROSCOPIC VIEWPOINT: SOFTWARE RELIABILITY IS THE PROBABILITY THAT THE USE OF THE SOFTWARE DOES NOT RESULT IN FAILURE OF A SYSTEM BY MORE THAN A TOLERABLE FREQUENCY. IN SHORT, SOFTWARE RELIABILITY IS THE PROBABILITY THAT THE SOFTWARE IS RELIABLE, UTILIZING A DEFINITION OF "RELIABLE SOFTWARE." B. SUBSYSTEM, DEVELOPER, OR MICROSCOPIC VIEWPOINT: SOFTWARE RELABILITY IS THE PROBABILITY THAT THE PROGRAM, ROUTINE, OR "MODULE" IS FAULT-FREE... SEE ALSO - RELIABLE SOFTWARE. (SET) (4) CODE POSSESSES THE CHARACTERISTIC RELIABILITY TO THE EXTENT THAT IT CAN BE EXPECTED TO PERFORM ITS INTENDED FUNCTIONS IN A SATISFACTORY MANNER. (DAN 239).

SOFTWARE RELIABILITY MEASURES

PROBABILITY MEASURES OF THE QUALITY WITH WHICH DESIGN REQUIREMENTS HAVE BEEN TRANSFORMED INTO SOFTWARE PROGRAMS. A TYPICAL MEASURE IS THE MEAN TIME TO FAILURE AS A FUNCTION OF OPERATING TIME. (DAN LD4)

SOFTWARE REQUIREMENTS DOCUMENT

(SRD) A DOCUMENT CHIEFLY GENERATED BY A CUSTOMER OR OTHER INITIATOR USED TO DISPLAY THE NEEDS, JUSTIFICATION, AND ESTIMATED COSTS ASSOCIATED WITH THE IMPLEMENTATION OF A DATA-PROCESSING CAPABILITY. SOME TECHNICAL MATERIAL, SUCH AS THAT NEEDED IN SUPPORT OF THE JUSTIFICATION OR ESTABLISHMENT OF

NEEDS, IS INCLUDED. DETAILED TECHNICAL REQUIREMENTS MAY BE APPENDED OR INCLUDED IN THE FUNCTIONAL REQUIREMENTS DOCUMENT (FRD) PORTION OF THE SRD. (DAN 1153)

SOFTWARE SCIENCE
A THEORY WHICH APPLIES THE SCIENTIFIC METHOD TO THE PROPERTIES AND STRUCTURE OF COMPUTER PROGRAMS. IT PROVIDES PRECISE OBJECTIVE MEASURES OF THE COMPLEXITY OF EXISTING SOFTWARE, PREDICTS THE LENGTH OF PROGRAMS, AND ESTIMATES THE AMOUNT OF TIME AN AVERAGE PROGRAMMER CAN BE EXPECTED TO USE TO IMPLEMENT A GIVEN ALGORITHM. THE THEORY DOES THIS BY COUNTING OPERATORS AND OPERANDS IN PROGRAMS. (DAN 231)

SOFTWARE SNEAK ANALYSIS
A FORMAL TECHNIQUE INVOLVING THE USE OF MATHEMATICAL GRAPH THEORY, ELECTRICAL SNEAK THEORY, AND COMPUTERIZED SEARCH ALGORITHMS WHICH ARE APPLIED TO A SOFTWARE PACKAGE TO IDENTIFY SOFTWARE SNEAKS. A SOFTWARE SNEAK IS DEFINED AS A LOGIC CONTROL PATH WHICH CAUSES AN UNWANTED OPERATION TO OCCUR OR WHICH BYPASSES A DESIRED OPERATION WITHOUT REGARD TO FAILURE OF THE HARDWARE SYSTEM TO RESPOND AS PROGRAMMED.(DAN 361)

SOFTWARE SNEAK CIRCUIT ANALYSIS
SNEAK CIRCUIT ANALYSIS (SCA) IS A TECHNIQUE BY WHICH A PROGRAM LOGIC STRUCTURE IS REDUCED TO A HARDWARE CIRCUITRY REPRESENTATION, AND THAT CIRCUITRY IS EXAMINED FOR A PREDEFINED SET OF FAULTS. THESE FAULTS INCLUDE OPEN-ENDED LOGIC, INFINITE LOOPS, BYPASS OF DESIRED LOGIC, UNNECESSARY LOGIC, MISSING LOGIC, UNUSED LOGIC, INCORRECT ADDRESSING, AND PROCEDURAL ERROR (E.G., UNINITIALIZED VARIABLES.)... THE SCA IS A SPECIALIZED FORM OF PEER CODE REVIEW LOOKING FOR THE CLASSES OF ERRORS (AND ONLY THOSE ERRORS) MENTIONED ABOVE. (SET)

SOFTWARE SPECIFICATION DOCUMENT
(SSD) THE PRINCIPAL PROGRAM DOCUMENTATION PRODUCED BY A DEVELOPMENT PROJECT, CONSISTING OF "AS-BUILT" FUNCTIONAL (SFS), PROGRAMMING (PS), AND TEST SPECIFICATIONS. (DAN 1153)

SOFTWARE STANDARDIZATION
EFFORTS TO STANDARDIZE SOFTWARE PACKAGES SO AS TO MAXIMIZE COST EFFECTIVENESS, MINIMIZE DEVELOPMENT TIME, AND INCREASE THE QUALITY OF EMBEDDED COMPUTER SYSTEMS.

SOFTWARE SYSTEM DEPENDABILITY
THE PROBABILITY THAT THE APPLICATION PROGRAM TOGETHER WITH ITS SUPERVISORY PROGRAM, DATA BASE AND HARDWARE WILL PERFORM IN ITS INTENDED ENVIRONMENT. THE ENVIRONMENT WILL INCLUDE ANOMALIES AND FAILURES, SUCH AS: 1) DEFICIENCIES IN REQUIREMENTS. 2) SOFTWARE DESIGN ERRORS (INCORRECT ALGORITHMS, WORD LENGTH PROBLEMS, TIMING PROBLEMS, ETC.) 3) SOFTWARE FAILURES 4) PROCESSOR ERRORS, 5) MEMORY ERRORS, 6) FAILURES IN THE COMMUNICATION NETWORK 7) FAILURES IN PERIPHERIAL DEVICES, 8) OPERATOR MISTAKES, 9) POWER FAILURES, 10) ENVIRONMENTAL FAILURES, 11) GRADUAL EROSION OF THE DATA BASE, 12) HARDWARE SATURATION (CPU, MEMORY, I/O CHANNELS) (DAN LD4)

SOFTWARE SYSTEMS
SEE SOFTWARE PRODUCT

SOFTWARE TESTING
THE PROCESS OF EXERCISING SOFTWARE IN AN ATTEMPT TO DETECT ERRORS WHICH EXIST IN THE CODE. SOFTWARE TESTING DOES NOT PROVE THAT A PROGRAM IS CORRECT. (DAN LD4)

SOURCE LANGUAGE DEBUG
SOURCE LANGUAGE DEBUG (SLD) IS THE PRACTICE OF DEBUGGING A PROGRAM VIA THE USER'S PROGRAMMING LANGUAGE AND VIA HUMAN-READABLE DATA VALUES. SLD MEANS TAKING JUMPS WHERE IDENTIFIERS LABEL THE CONTENTS, AND THE VALUES ARE AUTOMATICALLY FORMATTED ACCORDING TO A VARIABLE'S ATTRIBUTES. SLD MEANS BEING ABLE TO TRACE THE CHANGING VALUES OF ONE OR MORE VARIABLES, OR MONITORING THE LOGIC FLOW OF THE PROGRAM, VIA EASILY USED SOURCE LEVEL DEBUG STATEMENTS. SLD MEANS DYNAMIC DEBUG ENABLING AND DISABLING; AND SUPERIMPOSED OVER ALL OF THE ABOVE; THE ABILITY TO OPTIONALLY SUPPRESS DEBUG SOURCE STATEMENTS AT OR PRIOR TO COMPILATION WITHOUT HAVING TO MANUALLY REMOVE THEM. THE MOST OBVIOUS ADVANTAGE OF SLD IS PROGRAMMER CONVENIENCE. THE TASKS OF READING AND INTERPRETING MACHINE-LANGUAGE MAPS AND DUMPS ARE BANISHED, ALLOWING THE PROGRAMMER TO CONCENTRATE ON THE CLUES LEFT BEHIND BY HIS PROGRAM'S ERROR. THIS CONVENIENCE IS FORMIDABLE. THE PROGRAMMER IS RELIEVED OF SEVERAL DIFFICULT RESPONSIBILITIES, EACH OF WHICH HAS A MAJOR LEARNING CURVE ATTACHED TO IT: (1) LEARNING COMPUTER CHARACTERISTICS, SUCH AS THE INSTRUCTION SET AND FLOATING-POINT WORD FORMAT IN BINARY OR HEX; (2) LEARNING OPERATING SYSTEM CHARACTERISTICS, SUCH AS HOW TO READ CRYPTIC DUMP-RELATED MESSAGES; (3) LEARNING LOADER METHODOLOGY, IN ORDER TO UNDERSTAND THE COMPUTER'S MEMORY ALLOCATION; (4) LEARNING COMPILER IDIOSYNCRACIES, SUCH AS THE KIND OF OBJECT CODE GENERATED; AND (5) LEARNING JOB CONTROL LANGUAGE SEMANTICS IN ORDER TO SPECIFY NON-SLD TOOLS. ALTHOUGH A PROJECT MUST POSSESS THESE TALENTS AMONG ITS TEAM PARTICIPANTS, WITH SLD NOT EVERY PROGRAMMER NEEDS ALL OF THESE TALENTS. ADDITIONALLY, VIA SLD THE PROGRAMMER MAY PREPLAN HIS DEBUG ACTIVITIES AND PROVIDE FOR THEM DURING HIS JOB EXECUTION RATHER THAN DURING DEBUG ITSELF. (SET)

SOURCE PROGRAM
A COMPUTER PROGRAM EXPRESSED IN A PROGRAMMING LANGUAGE. (NASA)

SOURCE STATEMENTS
ALL STATEMENTS READABLE BY AND READ BY THE COMPILER. THIS INCLUDES EXECUTABLE STATEMENTS (E.G., ASSIGNMENT, IF GOTO,ETC.,), NONEXECUTABLE STATEMENTS (DIMENSION, REAL, ENDO, AND COMMENTS. (SEL) (2) (ISO) IN A PROGRAMMING LANGUAGE, A MEANINGFUL EXPRESSION THAT MAY DESCRIBE OR SPECIFY OPERATIONS AND IS COMPLETE IN THE CONTEXT OF THIS PROGRAMMING LANGUAGE. (ANSI-X3) (3) IN COMPUTER PROGRAMMING, A SYMBOL STRING OR OTHER ARRANGEMENT OF SYMBOLS. (ANSI-X3) (4) PROGRAMMING LANGUAGE AT THE SOURCE CODE LEVEL. (DAN 21)

SOURCE UNIT END DATE
THE DATE OF THE LAST UPDATE MADE TO A UNIT OF SOURCE CODE. (DAN 137)

SOURCE UNIT START DATA
THE DATE A UNIT OF SOURCE CODE WAS ENTERED INTO THE PROGRAM SUPPORT LIBRARY. (DAN 137)

SPARCS-2
SIMULATION PROGRAM FOR ASSESSING THE RELIABILITIES OF COMPLEX SYSTEMS

DEVELOPED AT OKLAHOMA STATE UNIVERSITY UNDER AIR FORCE CONTRACCT # F33615-74-C-4072 AND REWORKED UNDER F 33615-76-C-3094.

SPECIFICATION LANGUAGE
A LANGUAGE USED TO SPECIFY. SPECIFICATION LANGUAGES CAN BE USED TO SPECIFY DESIGNS, PROGRAMS, ETC.

SPECIFICATION OMISSION ERROR
AN ERROR FOR A UNIT OF SOURCE CODE ASSOCIATED WITH A PROGRAM DUE TO OMISSIONS IN THE PROGRAM SPECIFICATIONS. (DAN 137)

SPECIFICATION TOOLS AND TECHNIQUES
METHODS WHICH ASSIST IN CONSTRUCTION OF FORMAL SPECIFICATIONS. THESE METHODS CAN INCLUDE PROOF OF CORRECTNESS, USE OF A FIXED LANGUAGE, OR FIXED DISCIPLINES (DATA FLOW GRAPHS, V-GRAPHS), OR STATE MACHINES. (DAN 532)

SPECIFICATIONS
A DESCRIPTION OF THE INPUT, OUTPUT AND ESSENTIAL FUNCTION(S) TO BE PERFORMED BY A SYSTEM OR BY A COMPONENT OF THE SYSTEM. THE SPECIFICATION IS PRODUCED BY THE ORGANIZATION THAT IS TO DEVELOP THE SYSTEM; I.E. AT THE TOP LEVEL IT CAN BE THOUGHT OF AS THE CONTRACTOR'S INTERPRETATION OF THE REQUIREMENTS. VERY PRECISE SPECIFICATION - A COMPLETELY DEFINED DESCRIPTION OF THE INPUT, OUTPUT, AND FUNCTION OF A COMPONENT. THE IMPLEMENTOR OF A VERY PRECISE SPECIFICATION NEEDS TO MAKE FEW, IF ANY ASSUMPTIONS. IT IS ALMOST IMPOSSIBLE TO ARRIVE AT AN AMBIGUOUS INTERPRETATION OR MISUNDERSTANDING OF THE SPECIFICATIONS. PRECISE SPECIFICATION - THE INPUT, OUTPUT, AND FUNCTION OF THE COMPONENT ARE WELL DEFINED. THERE ARE UNDERLYING ASSUMPTIONS NOT SPECIFIED, BUT IT IS ASSUMED THAT ANY PROGRAMMER WORKING ON THE PROJECT, WITH EXPERIENCE ON A SIMILAR PROJECT, WILL UNDERSTAND THESE ASSUMPTIONS. IT IS POSSIBLE TO ARRIVE AT AN AMBIGUOUS INTERPRETATION OR MISUNDERSTANDING OF THE SPECIFICATIONS IF THE READER DOES NOT HAVE ENOUGH EXPERIENCE WITH THE PROBLEM OR DOES NOT OBTAIN FURTHER VERBAL COMMUNICATION. IMPRECISE SPECIFICATION - THE INPUT, OUTPUT, AND FUNCTION OF THE COMPONENT ARE LOOSELY DEFINED. MUCH OF WHAT IS REQUIRED IS ASSUMED RATHER THAN SPECIFIED. THE SPECIFICATION RELIES HEAVILY ON PROGRAMMER EXPERIENCE AND VERBAL COMMUNICATION IN ORDER TO GET AN UNAMBIGUOUS INTERPRETATION AND FULL UNDERSTANDING OF WHAT IS NEEDED. (SEL) SEE ALSO: FORMAL SPECIFICATIONS, ENGLISH (OR INFORMAL) SPECIFICATIONS FUNCTIONAL SPECIFICATIONS, PROCEDURAL SPECIFICATIONS. (2) THE TECHNICAL DEFINITION OF THE SYSTEM AND ITS PARTS. (DAN LD7)

SPECIFICATIONS DECOMPOSITION
DECOMPOSITION IN THIS INSTANCE REFERS TO BREAKING UP OF THE SPECIFICATIONS, ON BOTH A HORIZONTAL AND VERTICAL BASIS, TO DETERMINE ALL OF THE FUNCTIONS AND PROCESSES INVOLVED AND THEIR INTERRELATIONSHIPS.

SPECIFICATIONS DRIVEN
USING THE SPECIFICATIONS OF THE PROGRAM TO DETERMINE TEST DATA (E.G., TEST DATA IS GENERATED BY EXAMINING THE INPUT/OUTPUT REQUIREMENTS AND SPECIFICATIONS.). (SEL)

SPECIFICATIONS VERIFICATION
SPECIFICATIONS VERIFICATION IS THE PROCESS OF DETERMINING WHETHER OR NOT THE DESIGN SPECIFICATION FOR THE INDIVIDUAL COMPUTER PROGRAM MODULES REPRESENTS

A CLEAR, CONSISTENT, AND ACCURATE TRANSLATION OF THE COMPUTER PROGRAM REQUIREMENTS. SPECIFICATION VERIFICATION DOES NOT SEEK TO REDESIGN, BUT RATHER TO IDENTIFY INADEQUACIES. SEE ALSO - COMPUTER PROGRAM VERIFICATION

SPECIFIED PERFORMANCE REQUIREMENTS
A WRITTEN REQUIREMENT, FIGURE OF MERIT, OR PARAMETER WHICH QUALITATIVELY OR QUANTATITIVELY DEFINES SYSTEM PERFORMANCE. (DAN 31)

SREP
THE BALLISTIC MISSILE DEFENSE ADVANCED TECHNOLOGY CENTER IS SPONSORING AN INTEGRATED SOFTWARE DEVELOPMENT RESEARCH PROGRAM TO IMPROVE THE TECHNIQUES FOR DEVELOPING CORRECT, RELIABLE BMD SOFTWARE. THE SREP IS BEING DEVELOPED AS A PART OF THIS PROGRAM BY TRW DEFENSE AND SPACE SYSTEMS GROUP TO EXAMINE AND IMPROVE THE QUALITY OF REQUIREMENTS. (DAN 423)

STABILITY
STABILITY IS THE MEASURE OF LACK OF PERCEIVABLE CHANGE IN A SYSTEM, AT A GIVEN LEVEL OF THAT SYSTEM, IN SPITE OF SOME OCCURRENCE IN THE SYSTEM ENVIRONMENT WHICH WOULD NORMALLY BE EXPECTED TO CAUSE CHANGE. (DAN 781)

STACK
A DATA STRUCTURE THAT, TOGETHER WITH ITS ACCESS FUNCTIONS, MODELS OPERATIONS USED IN LAST-IN FIRST-OUT (LIFO) ALGORITHMS.

STANDARDIZATION
THE DEVELOPMENT OF CRITERIA FOR SOFTWARE TO MAXIMIZE RELIABILITY AND ESPECIALLY TRANSPORTABILITY.

STANDARDS
ANY SPECIFICATIONS THAT REFER TO THE METHOD OF DEVELOPMENT OF THE SOURCE PROGRAM ITSELF, AND NOT TO THE PROBLEM TO BE IMPLEMENTED (E.G., USING STRUCTURED CODE, AT MOST 100 LINE SUBROUTINES, ALL NAMES PREFIXED WITH SUBSYSTEM NAME, ETC.). (SEL) (2) PROCEDURES, RULES, AND CONVENTIONS USED FOR PRESCRIBING DISCIPLINED PROGRAM DESIGN (PROGRAM STRUCTURING, AND DATA STRUCTURING) AND IMPLEMENTATION. ARCHITECTURE AND PARTITIONING RULES, DOCUMENTATION CONVENTIONS, CONFIGURATION AND DATA MANAGEMENT PROCEDURES, ETC. ARE AMONG THOSE STANDARDS TO BE DISSEMINATED. (3) A DESIGN CRITERION. AN ENTITY CONFORMS TO A STANDARD IF THE ATTRIBUTE(S) DEFINED BY THE STANDARD APPLY TO THE ENTITY. (ABBOTT) (4) CONVENTIONS, GROUND RULES, GUIDELINES, PROCEDURES, AND SOFTWARE TOOLS EMPLOYED DURING THE SOFTWARE DEVELOPMENT PROCESS TO BENEFIT SOFTWARE DESIGN QUALITY, CODING QUALITY, SOFTWARE RELIABILITY, VIABILITY AND MAINTAINABILITY. (DAN 1201)

STANDARDS ENFORCER
A COMPUTER PROGRAM USED TO AUTOMATICALLY DETERMINE WHETHER PRESCRIBED PROGRAMMING STANDARDS AND PRACTICES HAVE BEEN ADHERED TO. THE PROGRAM CAN CHECK FOR VIOLATIONS TO STANDARDS SET FOR SUCH CONVENTIONS AS PROGRAM SIZE, COMMENTARY, STRUCTURE, ETC. (DAN 134)

START DATE
DATE INITIAL WORK ON PROJECT BEGAN. (SEL)

STATE DIAGRAM
A DEVICE USED TO DESCRIBE THE PROCESSING LOGIC IN TERMS OF STATES. A STATE

CAN BE DEFINED AS A POINT OF EQUILIBRIUM WHERE PROCESSING REMAINS DORMANT UNTIL AN EVENT OCCURS. (DAN 270)

STATE MACHINES
A MATHEMATICAL FRAMEWORK FOR DEFINING PRECISE SPECIFICATIONS OF COMPLEX SOFTWARE SYSTEMS.

STATEMENT
A UNIT OF A COMPUTER PROGRAM CONSISTING OF A MEANINGFUL ARRANGEMENT OF BASIC LANGUAGE ELEMENTS WHICH EXPRESSES A UNIFIED INSTRUCTION OR INFORMATION, ANALOGOUS TO A SENTENCE IN ENGLISH. (NASA) (2) THE ACT OR PROCESS OF STATING OR PRESENTING A SINGLE DECLARATION OR REMARK. (ANSI-X3H1)

STATIC ANALYSIS
STATIC ANALYSIS IS THE ANALYSIS OF A PROGRAM WITHOUT EXECUTING THE PROGRAM. SPECIFIC METHODOLOGIES INCLUDE DESK CHECKING, PEER CODE REVIEW, AND STRUCTURAL ANALYSIS.

STATIC REDUNDANCY
DUPLICATION OF PART OR ALL OF THE COMPONENTS (E.G. PROCESSOR, MAIN AND AUXILIARY MEMORIES, AND COMMUNICATIONS EQUIPMENT) SO THAT IN THE CASE OF FAILURE BY ONE UNIT ANOTHER CAN BE SWITCHED IN. (DAN 311)

STATISTICAL PREDICTION
THE COMPUTATION OF A CONFIDENCE FACTOR THAT INDICATES THE EFFECTIVENESS OF THE PROGRAMMING AND VERIFICATION PROCESS BY INSERTING ERRORS INTO THE SOFTWARE SYSTEM. (DAN 154)

STATISTICAL TEST MODELS
A MODEL WHICH RELATES DIFFERENT PROGRAM ERRORS TO THE INPUT DATA SET (OR SETS) WHICH EXCITE AND THUS DISPLAY A PARTICULAR ERROR. THE MODEL ALSO GIVES THE PROBABILITY THAT THESE ERRORS WILL CAUSE THE PROGRAM TO FAIL. (DAN 232)

STEPWISE REFINEMENT
STEP-WISE REFINEMENT IS THE PROCESS WHEREBY STEPS ARE TAKEN IN THE FOLLOWING ORDER: (1) THE TOTAL CONCEPT IS FORMULATED, (2) THE FUNCTIONAL SPECIFICATION IS DESIGNED, (3) THE FUNCTIONAL SPECIFICATION IS REFINED AT EACH INTERMEDIATE STEP WHERE THE INTERMEDIATE STEPS INCLUDE CODE OR PROCESSES REQUIRED BY THE PREVIOUS STEP, AND (4) FINAL REFINEMENTS ARE MADE TO COMPLETELY DEFINE THE PROBLEM. (SET) (2) THE PROCESS OF DEFINING DATA IN MORE AND MORE DETAIL AS THE NEED ARISES DURING THE PROGRAMMING PROCESS. (DAN 136) (3) THE DEFINING OF MORE GENERAL OPERATIONS IN TERMS OF MORE SPECIFIC, LOWER LEVEL OPERATIONS. THE DESIGN OF A PROGRAMMING SYSTEM THROUGH STEPWISE REFINEMENT IS CALLED TOP DOWN DESIGN. (ABBOTT)

STESD
A TOOL BEING DEVELOPED BY THE UNIVERSITY OF TEXAS AT AUSTIN WHICH; A) PROVIDES FOR DIRECT AND SYMBOLIC EXECUTION OF SPECIFICATIONS AS IF THEY WERE PROGRAMS, B) PROVIDES FOR PERFORMANCE AND COST ESTIMATES ON THE BASIS OF VARIOUS METHODS FOR SIMULATION AND CALCULATION AND C) PROVIDES FOR HEURISTIC JUDGEMENTS OF THE WAY IN WHICH THE DESIGN WHOLE IS DECOMPOSED INTO PARTS. (DAN 333)

STRENGTH

SEE COHESION

STRING PROCESSING
THIS INCLUDES COMPONENTS WHICH PERFORM OPERATIONS ON LISTS OF CHARACTERS. NORMALLY, WE THINK OF THIS CLASS TO INCLUDE FUNCTIONS OF COMPILERS, HASH CODE STRING HOOK-UP AND ARRAY COMPARISONS. (SEL)

STRIPED MODULE
A NAMED MODULE IN THE PROGRAM PROCEDURAL DESIGN, SO CALLED BECAUSE OF THE METHOD USED TO DENOTE SUCH MODULES ON A FLOWCHART. STRIPING OF A FLOWCHART SYMBOL SIGNIFIES THAT A DETAILED REPRESENTATION IS EITHER LOCATED ELSEWHERE IN THE SAME SET OF FLOWCHARTS (HORIZONTAL STRIPING), OR ELSE AT A REFERENCED LOCATION (VERTICAL STRIPING). (DAN 1153)

STRONG TYPING
A SOFTWARE DESIGN AND CODING CRITERION. A SYSTEM CONFORMS TO THE CRITERION OF STRONG TYPING TO THE EXTENT THAT DIFFERENT DATA TYPES ARE DEVELOPED AND USED FOR SEPARATE SORTS OF DATA OBJECTS. A SYSTEM FAILS TO CONFORM TO THE CRITERION OF STRONG TYPING TO THE EXTENT THAT ONE DATA TYPE IS USED ON THE SAME LEVEL OF ABSTRACTION TO ENCODE THE VALUES OF DATA OBJECTS OF A DIFFERENT DATA TYPE. (SEE ENCODING) EXAMPLE: AN OBJECT OF THE DATA TYPE BOOLEAN HAS THE POSSIBLE VALUES TRUE AND FALSE. THESE VALUES MAY BE ENCODED AS THE INTEGER VALUES 1 AND 0. TO THE EXTENT THAT ENCODING IS USED ON THE SAME LEVEL OF OBSTRUCTION IN PLACE OF THE VALUES TRUE AND FALSE, THE SYSTEM IS NOT IN CONFORMANCE TO THE CRITERION OF STRONG TYPING. TO THE EXTENT THAT TRUE AND FALSE, USED ON ONE LEVEL OF ABSTRACTION, ARE IMPLEMENTED, ON A LOWER LEVEL OF ABSTRACTION, AS INTEGER VALUES 1 AND 0, THE SYSTEM IS IN CONFORMANCE TO THE CRITERION OF STRONG TYPING. (ABBOTT)

STRUCT
AN ALGORITHM FOR AUDITING PROGRAMS FOR COMPLIANCE WITH A STRUCTURED PROGRAMMING STANDARD. (DAN 260)

STRUCTURAL COMPLEXITY
A MEASURE OF THE DEGREE OF SIMPLICITY OF RELATIONSHIPS BETWEEN SUBSYSTEMS. (DAN 781) (2) SEE ALSO COMPLEXITY, LOGICAL COMPLEXITY. (3) SYNONOMOUS WITH MODULARITY (4) THE DEGREE OF COUPLING AMONG MODULES OF A COMPUTER PROGRAM.(NASA)

STRUCTURAL INTEGRATION
THE COMBINATION OF RELATED HARDWARE, IRRESPECTIVE OF FUNCTIONAL APPLICATIONS INTO A SYSTEM ARCHITECTURE WHICH PROVIDES COST OF PERFORMANCE BENEFITS. (NASA)

STRUCTURAL MODEL
MODEL THAT DEPICTS FUNCTIONAL RELATIONSHIPS AMONG ACTIVITIES AND/OR PROCESSES STRUCTURALLY, WHETHER GRAPHICAL OR OTHERWISE. (DAN 255)

STRUCTURE
MAY PERTAIN TO THE MANNER OR FORM IN WHICH SOMETHING IS CONSTRUCTED OR MAY REFER TO THE ACTUAL SYSTEM AS CONSTRUCTED. DESCRIPTIONS OF STRUCTURE FOCUS AN INTERRELATION OF THE VARIOUS PARTS AS DOMINATED BY THE GENERAL CHARACTER OR FUNCTION OF THE WHOLE. DESIGNING STRUCTURE IS A PROCESS OF IDENTIFYING, ANALYZING, AND SELECTING AMONG ALTERNATIVES WITHIN DESIGN CATEGORIES. (DAN

1153)

STRUCTURE CHARTS
A GRAPHICAL TECHNIQUE WHICH ILLUSTRATES THE RELATIONSHIPS BETWEEN THE COMPONENTS OF A SOFTWARE SYSTEM.

STRUCTURE DRIVEN
USING THE STRUCTURE OF THE PROGRAM TO DETERMINE TEST DATA (E.G. GENERATING DATA TO ENSURE THAT EACH BRANCH OF A PROGRAM IS EXECUTED AT LEAST ONCE.) (SEL)

STRUCTURE FLAG
A FLAG INTRODUCED INTO AN OTHERWISE UNSTRUCTURED PROGRAM TO PERMIT STRUCTURED CONTROL FLOW. (DAN 1153)

STRUCTURE GRAPH
A GRAPHICAL REPRESENTATION SHOWING THE CONTROL CONNECTIONS BETWEEN NAMED MODULES. THE "TOP" NODE OF THE GRAPH REPRESENTS THE TOP-LEVEL MAIN PROGRAM PROCEDURE; LINES FROM THE TOP NODE TO OTHER NODES SIGNIFY THAT THE CORRESPONDING NAMED MODULES APPEAR AS INVOCATIONS IN THE TOP-LEVEL PROGRAM PROCEDURE, ETC. (DAN 1153)

STRUCTURE OF DATA
THE ORGANIZATION OF A COMPOSITE DATA ITEM CONSISTING OF SEVERAL VARIABLES OR OTHER ARRAY ITEMS. EXAMPLES OF SUCH COMPOSITE DATA ITEMS ARE ARRAYS (BOTH SINGLY AND MULTIPLY DIMENSIONED), STRINGS, COMPLEX, VARIABLES AND CONSTANTS, RECORDS ON A DISK FILE (EACH RECORD CONTAINING SEVERAL WORDS), MULTIPLE-WORD ENTRIES IN A TABLE, ETC. (SEL)

STRUCTURED CODE
THE LANGUAGE SUPPORTS STRUCTURED CONTROL STRUCTURES (E.G., A FORTRAN PREPROCESSOR). (SEL) (2) A DESIGN AND CODING CRITERION. A PROGRAM SATISFIES THE CRITERION OF STRUCTURED CODE IF ITS OPERATIONS ARE ORGANIZED AS A CONTROL SEGMENTS: IF THE ORDER IN WHICH ITS OPERATIONS ARE PERFORMED IS DETERMINED BY CONTROL STRUCTURES AND NOT JUST CONTROL STATEMENTS. (ABBOTT)

STRUCTURED DESIGN
STRUCTURED DESIGN IS A SET OF TECHNIQUES FOR REDUCING THE COMPLEXITY OF LARGE NEW PROGRAMS BY DIVIDING THEM INTO INDEPENDENT MODULES. WORKING WITH SEPARATE PIECES PERMITS THE PROGRAMMER TO CODE, DEBUG, TEST, AND MODIFY A FUNCTIONAL MODULE WITH MINIMAL EFFECT ON OTHER MODULES OF THE ENTIRE SYSTEM. CONCENTRATING EFFORT IN THIS WAY ENHANCES EFFICIENCY AND QUALITY AND REDUCES BUGS. MOREOVER, TO THE EXTENT THAT THE INDEPENDENT MODULES ARE PORTABLE, FURTHER SYSTEMS CAN BE DEVELOPED WITH LESS NEED FOR NEW CODE. (DAN 227)

STRUCTURED NARRATIVE
THE PROCEDURE OF WRITING A DESIGN SPECIFICATION AS A SET OF STEPS EXPLAINING THE OPERATION OF THE PROGRAM. EACH STEP USES A FORMAL CONSTRUCT WITH DATA PERTINENT TO THE PROGRAM. (DAN 1201)

STRUCTURED PROGRAM
A PROGRAM CONSTRUCTED OF A BASIC SET OF CONTROL LOGIC FIGURES WHICH PROVIDE AT LEAST THE FOLLOWING: SEQUENCE OF TWO OPERATIONS, CONDITIONAL BRANCH TO ONE OF TWO OPERATIONS AND RETURN AND REPETITION OF AN OPERATION. A

STRUCTURED PROGRAM HAS ONLY ONE ENTRY AND ONE EXIT POINT. IN ADDITION, A PATH WILL EXIST FROM THE ENTRY TO EACH NODE AND FROM EACH NODE TO THE EXIT. (DAN 140)

STRUCTURED PROGRAM SEGMENTS

A COMBINATION OF PROGRAM STEPS AND CALLS TO LOWER-LEVEL PROGRAMS. (DAN [illegible]) SEE ALSO STRUCTURED SEGMENT.

STRUCTURED PROGRAMMING

STRUCTURED PROGRAMMING IS THE ACTIVITY OF PROGRAMMING WITH A LIMITED SET OF CONSTRUCTS. THE KEY CONSTRUCTS IN STRUCTURED PROGRAMMING ARE: (A) EACH PROGRAM IS ALLOWED ONLY ONE ENTRY AND ONE EXIT. (B) ONLY THREE BASIC CONTROL STRUCTURES ARE SUFFICIENT: DO-WHILE, IF-THEN-ELSE, ELSE, AND SEQUENCE. (C) OTHER CONSTRUCTS ARE SOMETIMES ALLOWED, THE MOST POPULAR ONES BEING DO-UNTIL AND CASE. (D) THE RESTRICTED CONSTRUCTS ARE OFTEN AUGMENTED WITH THE FOLLOWING PRACTICES: * HIERARCHICAL AND MODULAR BLOCK STRUCTURES * LIMITS ON SIZE OF MODULES * THE PROPER INDENTATIONS AND FORMATTING OF INSTRUCTIONS AND DECLARATIONS FOR EASE OF READING. (SET) (2) TO ENHANCE READABILITY AND MAINTAINABILITY, A PROGRAM IS STRUCTURED SO THAT LOGIC FLOW PROCEEDS FROM BEGINNING TO END WITHOUT ARBITRARY BRANCHING. THIS APPROACH IS BASED ON THE THEOREM THAT ANY PROGRAM WITH ONE ENTRY AND ONE EXIT CAN BE RECONSTRUCTED FROM ONLY THREE CONTROL STRUCTURES: SEQUENCE, IF-THEN-ELSE, AND DO-WHILE. IT IS ANALOGOUS TO THE FORMATION OF COMPLEX LOGIC FUNCTIONS FROM ONLY AND NOT BUILDING BLOCKS. (3) THE TECHNIQUE USED IN STRUCTURED PROGRAMMING (LIMITED NUMBER OF LOGIC STRUCTURES, TOP-DOWN DEVELOPMENT, ETC.) MAKE IT EASIER TO DEVELOP AND VERIFY COMPLETED COMPUTER PROGRAMS. AIDS THAT AID IMPLEMENT THESE TECHNIQUES FOLLOW. 1) AUTOMATED LANGUAGE RESTRUCTURER - A COMPUTER PROGRAM USED TO RESTRUCTURE SOURCE PROGRAMS INTO AN ACCEPTABLE STRUCTURED FORM. 2) LANGUAGE ENHANCER - A COMPUTER PROGRAM USED TO PROVIDE EXISTING HIGH-ORDER LANGUAGE COMPILERS WITH ACCEPTABLE LANGUAGE CONSTRUCTS FOR STRUCTURED PROGRAMMING. LANGUAGE ENHANCERS INCLUDE BOTH THE PREPROCESSOR AND MACRO MECHANIZATIONS PRESENTLY IN USE. 3) SOURCE CODE INDENTER - A COMPUTER PROGRAM USED TO AUTOMATICALLY INDENT SOURCE CODE LISTINGS TO AID IN THEIR READABILITY. 4) PRODUCTION SUPPORT LIBRARIES - A FORM OF REFERENCE LIBRARY USED TO RECORD AND STORE PROGRAMMING DATA. (DAN 134) (4) (SET) THE PROCESS OF DEVELOPING STRUCTURED PROGRAMS. ASSOCIATED WITH STRUCTURED PROGRAMMING ARE CERTAIN PRACTICES SUCH AS INDENTATIONS OF SOURCE CODE TO REPRESENT LOGIC LEVELS, THE USE OF INTELLIGENT DATA NAMES AND DESCRIPTIVE COMMENTARY. (DAN 140) (5) A PROGRAMMING DISCIPLINE PROVIDING A MEANS OF EXPRESSING A SYSTEM DESIGN THAT ENSURES A TESTABLE AND UNDERSTANDABLE IMPLEMENTATION AND THAT ENFORCES SIMPLE AND WELL-DEFINED CONNECTIONS BETWEEN PROGRAM MODULES. THE PROGRAMMING DISCIPLINE USES THE REPEATED APPLICATION OF A SMALL NUMBER OF BASIC CONTROL STATEMENTS TO FORM SIMPLE PROGRAM CONSTRUCTS THAT REPRESENT LARGE AND COMPLEX PROGRAMS. THE PROCESS OF CREATING THE PROGRAM MODULES INCLUDES: MAKING LOCAL OR TACTICAL PROGRAMMING DECISIONS WITHIN THE DESIGNED MODULE; WRITING PROGRAM SEGMENTS THAT REPRESENT THESE DECISIONS; AND, INTEGRATING PROGRAM SEGMENTS INTO A UNIT CORRESPONDING TO A SYSTEM MODULE. (DAN LD7) (6) STRUCTURED PROGRAMMING IS STRICTLY MORE OF A PROGRAM ORGANIZATION DISCIPLINE THAN A DESIGN TECHNIQUE, BUT IT CAN BE USED TO SIGNIFICANTLY ENHANCE MOST OF THE AVAILABLE DESIGN TECHNIQUES. IT IS BASICALLY A SET OF STANDARDS FOR ORGANIZING THE CONTROL STRUCTURE OF A SET OF COMPUTER PROGRAMS. (DAN776)

STRUCTURED PROGRAMMING LANGUAGE

A LANGUAGE THAT SUPPORTS THE CONCEPTS OF STRUCTURED PROGRAMMING AND PERMITS CLOSER INTEGRATION OF MANAGEMENT AND PRODUCTION TOOLS. (DAN 102) SEE ALSO SIMULATING CONSTRUCTS.

STRUCTURED PROGRAMMING TECHNOLOGY
A TERM WHICH COLLECTIVELY REFERENCES THE P.D. WING UNIT'S PROGRAM DESIGN LANGUAGE, ITS PROGRAMMING SUPPORT LIBRARY, AND THE TEAM STRUCTURED PROGRAMMING (DAN 130) SEE ALSO STRUCTURED PROGRAMMING TECHNIQUES

STRUCTURED SEGMENT
A LOGICALLY COMPLETE SET OF [illegible] INSTRUCTIONS OR DECLARATIVES OF WHICH STRUCTURED PROGRAMMING IS MADE. IN ADDITION TO OR IN PLACE OF EXECUTABLE INSTRUCTIONS, A STRUCTURED SEGMENT MAY INCLUDE NON-EXECUTABLE INSTRUCTIONS SUCH AS DATA DECLARATIVES AND DESCRIPTIVE COMMENTARY. (DAN 130) SEE ALSO STRUCTURED PROGRAM SEGMENTS.

STRUCTURED SOURCE CODE LISTING
A LISTING FOR A [illegible] PROGRAM [illegible] WHICH IS THE FOLLOWING SET OF [illegible] WITH THE FIRST AND LAST STRUCTURED SEGMENT [illegible] AS THE [illegible] SEGMENTS AS USED IN THE SOURCE PROGRAMMING LANGUAGE. [illegible] CONTAINS ALL REMAINING SETS OF SEGMENTS. THE [illegible] SEGMENTS ARE ALPHABETICAL BY NAME, AS IN SECTION 1. EACH SET OF SEGMENTS IS ARRANGED AS USED IN THE SOURCE PROGRAMMING LANGUAGE. [illegible] CONTAINS THE [illegible] THE STRUCTURED SEGMENTS. (DAN 130)

STRUCTURED WALK-THROUGH
A GENERIC NAME GIVEN TO A SET OF TECHNIQUES OF [illegible] VALIDATION, DESIGN VERIFICATION, AND [illegible] WITH DIFFERENT EMPHASES AND EACH OF WHICH, AT DIFFERENT TIMES IN THE SOFTWARE DEVELOPMENT CYCLE. (DAN 144) (2) THE USE OF THE WALK-THROUGH AS A METHOD FOR FORMAL DESIGN REVIEW, IN THE PDL DESIGN SYSTEM WITH SUPPORT AND VERIFICATION, AT THE BEGINNING TAKES THE INITIATIVE TO PLAN AND RUN THE SESSION. MANAGEMENT DOES NOT ATTEND, THUS ENCOURAGING A MORE OPEN ATMOSPHERE. 3) IN THE NEW DESIGN AND PROGRAMMING PRACTICES [illegible] MAINTAINABLE SYSTEMS THE DESIGNER OR PROGRAMMER WALKS THROUGH [illegible] THE DESIGN OR CODE WITH A [illegible] REVIEWER OR REVIEWERS [illegible] WHILE THE [illegible] (DAN 205)

STRUCTUREDNESS
CODE POSSESSES THE CHARACTERISTIC "STRUCTUREDNESS" TO THE EXTENT THAT IT POSSESSES A DEFINITE PATTERN OF ORGANIZATION OF ITS INTERDEPENDENCIES. THIS IMPLIES THAT EVOLUTION OF THE PROGRAM DESIGN HAS PROCEEDED IN AN ORDERLY AND SYSTEMATIC MANNER, AND THAT STANDARD CONTROL STRUCTURES HAVE BEEN FOLLOWED IN DEFINING THE PROGRAM, ETC. (DAN 205)

STRUCTURE-CHANGING PROGRAMS
PROGRAMS THAT CAN RESET THE [illegible] OF A [illegible], CHANGE THE ELEMENTS OF AN ARRAY, OR ALTER THE STRUCTURE OF A [illegible] DATA OBJECT. (DAN 205)

STUB
A "DUMMY" SOFTWARE ELEMENT USED IN PLACE OF AN EXPECTED FUNCTIONAL ELEMENT UNTIL THE EXPECTED ELEMENT BECOMES AVAILABLE. A STUB CAN SATISFY INTERFACE REQUIREMENTS, TRACE EXECUTION FLOW, AND MEET PROGRAM BEHAVIOR BY CONSUMING

CPU TIME, OCCUPYING MEMORY SPACE, AND USING MASS MEMORY. (DAN 1201)

SUBMARINE APPLICATION
INDEXING TERM. REFERS TO SOFTWARE USED AS A COMPONENT IN A SUBMARINE OR TO THE USE OF SOFTWARE AS A TOOL IN THE DESIGN OR CONSTRUCTION OF SUBMARINES.

SUBMODULE
A MODULE APPEARING WITHIN A MODULE OR INVOKED BY A MODULE. ON A FLOWCHART, THE PROCEDURE APPEARING WITHIN OR REFERRED TO (E.G., INVOKED BY) ANY CHARTED SYMBOL. (DAN 1153)

SUBPROGRAM
A SUBPROGRAM IS A COMPUTER PROGRAM THAT CAN BE PART OF ANOTHER PROGRAM. IT IS USUALLY A PROGRAM WHICH CAN BE A SUBROUTINE OR A FUNCTION THAT IS INVOKED BY A CALL STATEMENT. ALSO SEE - SUBROUTINE (SET) (2) A COLLECTION OF PROGRAM ELEMENTS WHICH TOGETHER PROVIDE A FUNCTION OR RELATIVELY INDEPENDENT FUNCTIONS WITH RESPECT TO THE WHOLE PROGRAM. (DAN 1201) COMPARE WITH MODULE.

SUBROUTINE
A SUBROUTINE IS A SUBPROGRAM THAT DOES NOT RETURN A VALUE ASSOCIATED WITH ITS NAME WHEN INVOKED. A SUBROUTINE CAN BE OPEN OR CLOSED. AN OPEN SUBROUTINE IS INSERTED AS IN-LINE CODE AT EACH PLACE IT IS USED. A CLOSED SUBROUTINE CAN BE STORED AT ONE PLACE AND CAN BE LINKED TO ONE OR MORE CALLING ROUTINES THROUGH PARAMETERS PASSED DURING EACH CALL. SEE ALSO SUBPROGRAM (SET)

SUBSYSTEM
A SUBDIVISION OF A SOFTWARE SYSTEM WITH A MEANINGFUL FUNCTION AND PURPOSE WHICH IS COMPRISED OF ONE OR MORE PROGRAMS. A SUBSYSTEM NAME IDENTIFIES THE TOP UNIT OF A TREE STRUCTURE. (DAN 1201) (2) A COLLECTION OF PROGRAMS WHICH TOGETHER PROVIDES A MAJOR FUNCTION AND IS RELATIVELY INDEPENDENT FROM THE SUBSYSTEM. (DAN 1201)

SUPERVISORY PROGRAM
(ISO) A COMPUTER PROGRAM, USUALLY PART OF AN OPERATING SYSTEM, THAT CONTROLS THE EXECUTION OF OTHER COMPUTER PROGRAMS AND REGULATES THE FLOW OF WORK IN A DATA PROCESSING SYSTEM. SYNONYMOUS WITH EXECUTIVE PROGRAM, SUPERVISOR. (ANSI-X3)

SUPPORT SOFTWARE
ALL PROGRAMS USED IN THE DEVELOPMENT AND MAINTENANCE OF THE DELIVERED OPERATIONAL PROGRAMS AND TEST/MAINTENANCE PROGRAMS. SUPPORT PROGRAMS INCLUDE, BUT ARE NOT LIMITED TO: A) COMPILERS, ASSEMBLERS, EMULATORS, BUILDERS, AND LOADERS REQUIRED TO GENERATE MACHINE CODE AND TO COMPOSE SUBPROGRAMS OR COMPONENTS INTO A COMPLETE COMPUTER PROGRAM. B) DEBUGGING PROGRAMS C) STIMULATION AND SIMULATION PROGRAMS USED IN OPERATOR TRAINING SITES, D) DATA ABSTRACTION AND REDUCTION PROGRAMS APPLICABLE TO OPERATIONAL PROGRAMS. E) TEST PROGRAMS USED IN DEVELOPMENT OF OPERATIONAL PROGRAMS. F) PROGRAMS USED FOR MANAGEMENT CONTROL, CONFIGURATION MANAGEMENT OR DOCUMENT GENERATION AND CONTROL DURING DEVELOPMENT. (DAN 384) (2) A COMPUTER PROGRAM WHICH FACILITATES THE DESIGN, DEVELOPMENT, TESTING, ANALYSIS, EVALUATION, OR OPERATION OF OTHER COMPUTER PROGRAMS.(NASA) (3) SOFTWARE TOOLS USED BY PROJECT PERSONNEL FOR SOFTWARE DESIGN, DEBUGGING, TESTING, VERIFICATION, AND MANAGEMENT. (DAN 1201)

SUPPORT TOOLS
THE SET OF SOFTWARE TOOLS AND PROCEDURES USED AS AIDS IN SOFTWARE DEVELOPMENT. (DAN 1201)

SUSTAINING ENGINEERING
SOFTWARE-RELATED ACTIVITIES IN THE POST-DELIVERY PERIOD, PRINCIPALLY SUPPORTIVE IN FORM, WHICH KEEP THAT SOFTWARE OPERATIONAL WITHIN ITS FUNCTIONAL SPECIFICATIONS; E.G., REPAIRING FAULTS, CORRECTING DOCUMENTATION, REMOVING LIENS, AND ESTIMATING COSTS AND OTHER RESOURCES REQUIRED FOR SUCH TASKS. THE HOLDING OR KEEPING OF SOFTWARE IN A STATE OF EFFICIENCY OR VALIDITY DESPITE INTERFACE FLUCTUATIONS IN SYSTEM, SUBSYSTEM, OR APPLICATIONS CAPABILITIES. (DAN 1153)

SYMBOLIC EXECUTION
"INSTEAD OF EXECUTING A PROGRAM ON A SET OF SAMPLE INPUTS, A PROGRAM IS SYMBOLICALLY EXECUTED FOR A SET OF CLASSES OF INPUTS LEADING TO SYMBOLIC VALUES WHICH DESCRIBE THE RELATION BETWEEN INPUT AND OUTPUT." (DAN 281)

SYMBOLIC MACHINE LANGUAGE
MACHINE LEVEL LANGUAGE UTILIZING SYMBOLS AND/OR MNEMONICS TO REPRESENT PARTICULAR SEQUENCES OF 0'S AND 1'S. (NASA)

SYNCHRONIZATION
THE PROCESS OF SETTING UP A MECHANISM TO PERFORM OPERATIONS OR PROCEDURES IN A TIME OR SEQUENCE HEIRARCHY. (DAN 210) (2) THE SCHEME BY WHICH ARBITRATION CONSTRAINS THE ORDERING OF OPERATIONS ON SHARED RESOURCES AMONG CONCURRENT PROCESSES IN TIME SO AS TO ENABLE CONSISTENCY IN THE PROGRAM BEHAVIOR. (DAN 1153)

SYNONYM
AN ADDITIONAL NAME IN THE SAME LANGUAGE BY WHICH AN ITEM IS KNOWN. (ANSI-X3H1) SEE ALSO ALIAS.

SYNTAX
THE PART OF A GRAMMAR DEALING WITH THE WAY IN WHICH ITEMS IN A LANGUAGE ARE ARRANGED. (ANSI-X3H1) (2) THE SET OF RULES THAT DEFINES THE VALID INPUT STRINGS (SENTENTIAL FORMS) OF A COMPUTER LANGUAGE AS ACCEPTED BY ITS COMPILER (OR ASSEMBLER). THEREFORE, THE STRUCTURE OF EXPRESSIONS IN A LANGUAGE, OR THE RULES GOVERNING THE STRUCTURE OF A LANGUAGE. (DAN 1153)

SYNTHESIZERS
THE SYNTHESIZER IS A PROGRAM THAT GENERATES TEST CASE PROGRAMS, FROM A GIVEN GRAMMAR, THAT MEET SYNTACTIC REQUIREMENTS, OR CONSTRAINTS. (DAN 235)

SYSTEM
(ISO) IN DATA PROCESSING, A COLLECTION OF MEN, MACHINES, AND METHODS ORGANIZED TO ACCOMPLISH A SET OF SPECIFIC FUNCTIONS. (2) A SYSTEM IS A SET OF RELATED COMPONENTS OR SUBSYSTEMS. THE COMPONENTS MAY INCLUDE COMPUTER HARDWARE, A PROJECT ORGANIZATION (IN TERMS OF PEOPLE AND METHODS), A COMPUTER PROGRAM, CODES AND DATA (AN INPUT LANGUAGE). ETC. (DAN 781) (3) CONSISTS OF MORE THAN ONE SOFTWARE SUBSYSTEM AND PROVIDES A SOLUTION TO A PROBLEM. (DAN 137) (4) A COLLECTION OF HUMANS, MACHINES, AND METHODS, ORGANIZED TO ACCOMPLISH A PURPOSE. (ANSI-X3H1) (5) A SET OR ARRANGEMENT OF SOFTWARE OR HARDWARE SO RELATED OR CONNECTED AS TO FORM A UNITY CAPABLE OF

ACHIEVING THE GOALS SPECIFIED IN ITS DESIGN. (DAN 1201)

SYSTEM ACQUISITION MANAGEMENT
REFERS TO A MANAGEMENT APPROACH TO ACQUIRING COMPUTER SYSTEMS WHICH ENCOMPASSES THE WHOLE SYSTEM, WITH EMPHASIS ON THE SOFTWARE, FROM THE INITIAL CONCEPT FORMULATION TO THE SUPPORT OF THE OPERATIONAL SYSTEM.

SYSTEM ARCHITECTURE
SYNONOMOUS WITH COMPUTER ARCHITECTURE.

SYSTEM DEPENDENCIES
DATA, PARAMETERS, OR EQUIPMENT NECESSARY FOR THE SYSTEM TO FUNCTION PROPERLY.

SYSTEM DESIGN
THE PROCESS OF TRANSFERRING THE DESIGN REQUIREMENTS INTO AN OVERALL SYSTEM STRUCTURE THAT SUPPORTS PROGRAMS SATISFYING THOSE REQUIREMENTS. INCLUDES ALL ACTIVITIES CONCERNED WITH TRANSFORMING FUNCTIONAL REQUIREMENTS/SPECIFICATIONS INTO A STRUCTURED PROGRAMMING SYSTEM CAPABLE OF SATISFYING THOSE REQUIREMENTS. (DAN LD7) (2) TRANSLATION OF THE REQUIREMENTS INTO A DESCRIPTION OF ALL THE COMPONENTS NECESSARY TO IMPLEMENT THE SYSTEM. (DAN 773)

SYSTEM DESIGN LANGUAGES
A DESIGN TOOL FOR SPECIFYING ABSTRACT MACHINE ARCHITECTURES. (ABBOTT)

SYSTEM ENGINEERING LANGUAGE
A MULTI-PURPOSE LANGUAGE WHICH CAN BE USED FOR REQUIREMENTS AND DESIGN SPECIFICATION, AUTOMATIC SIMULATION MODEL GENERATION, AUTOMATED DESIGN ANALYSIS AND VERIFICATION.

SYSTEM INTEGRATION
THE PROCESS OF COMBINING SYSTEM COMPONENTS TOGETHER TO PRODUCE THE TOTAL SYSTEM. (DAN 318) (2) THE PROCESS OF COMBINING PHYSICALLY, ELECTRONICALLY, AND FUNCTIONALLY ALL ELEMENTS SPECIFIED FOR A SYSTEM, USUALLY COMPOSED OF BOTH HARDWARE AND SOFTWARE. FOR EXAMPLE, SYSTEM INTEGRATION IS PERFORMED AT AN EVALUATION FACILITY BEFORE ACCEPTANCE TESTING MAY BEGIN. (DAN 1201)

SYSTEM RELIABILITY
A MEASURE OR INDICATION OF THE SUCCESS WITH WHICH THE SYSTEM PROVIDES THE SERVICE SPECIFIED. (DAN 236) SYSTEM HERE REFERS TO BOTH THE HARDWARE AND SOFTWARE AS A PACKAGE.

SYSTEM SIMULATIONS
COMPUTER SYSTEM SIMULATION IS A TECHNIQUE USED TO PREDICT SYSTEM PERFORMANCE BY EXERCISING A MODEL OF THE SYSTEM HARDWARE/SOFTWARE OVER TIME. SIMULATION BASED ON WELL-PLANNED EXPERIMENTS REPRESENTATIVE OF THE REAL-WORLD ENVIRONMENT WILL PRODUCE RESULTS THAT HELP VERIFY AND IMPROVE SYSTEM PERFORMANCE. THE SIMULATIONS ARE ALSO USED TO HELP PREDICT HOW THE SYSTEM WILL REACT TO ALTERNATIVE LOADS WITH MODIFIED CONFIGURATIONS. SPECIFIC LANGUAGE SYSTEMS SUCH AS ECSS, CSS, SCERT, AND SAM HAVE BEEN DEVISED TO ACT AS AIDS TO IMPLEMENTATION. (DAN 134)

SYSTEM SIZE

TOTAL NUMBER OF MACHINE WORDS NEEDED FOR ALL INSTRUCTIONS GENERATED ON THE PROJECT PLUS SPACE FOR DATA, LIBRARY ROUTINES AND OTHER CODE. THIS IS THE TOTAL SIZE OF THE SYSTEM WITHOUT USING ANY OVERLAY STRUCTURE. (SEL)

SYSTEM SPECIFICATION VERIFICATION
SEE COMPUTER PROGRAM VERIFICATION

SYSTEM STRUCTURING
PLACING CONSTRAINTS UPON THE INTERRELATIONSHIPS BETWEEN THE COMPONENTS OF A SYSTEM. (DAN 236)

SYSTEM SURVIVAL PROBABILITY
SYNONOMOUS WITH INTEGRITY PROBABILITY (DAN 781)

SYSTEM TESTING
SYSTEM TESTING IS THE PROCESS OF TRYING TO FIND DISCREPANCIES BETWEEN THE SYSTEM AND ITS ORIGINAL OBJECTIVES. (DAN 286).

SYSTEM VALIDATION
THE PROCESS OF CHECKING EQUIPMENT CONFORMITY WITH SPECIFICATIONS. (DAN 258)

SYSTEM VERIFICATION
ALL ACTIVITIES CONCERNED WITH ASCERTAINING THAT THE SYSTEM PERFORMS AS THE CUSTOMER INTENDED. (DAN LD7)

SYSTEMS RELATED SOFTWARE
BY SYSTEMS RELATED SOFTWARE, WE INCLUDE ANY PACKAGE DESIGNED TO AFFECT, MODIFY, EXTEND OR CHANGE THE 'NORMAL' AVAILABLE PROCESSING PROCEDURE OF THE OPERATING SYSTEM. THIS COULD INCLUDE SUCH COMPONENTS AS ERROR TRACING, OR EXTENDED I/O SUCH AS DAIO. (SEL)

TABLE
IN PROGRAMMING THE TERM TABLE MAY BE USED SYNONOMOUSLY WITH "ARRAY", BUT IS TYPICALLY DISTINGUISHABLE FROM THE ARRAY IN BEING UNIDIMENSIONAL OR NON-MATHEMATICS ORIENTED.

TABLE HANDLER
THIS INCLUDES COMPONENTS WHICH ARE SPECIFICALLY DESIGNED TO GENERATE OR INTERPRET INFORMATION WHICH IS IN A TABLE FORMAT SUCH AS THE GENERALIZED TELEMETRY PROCESSOR. (SEL)

TACTICS
A LARGE INTERACTIVE STATISTICAL ANALYSIS AND MODELING SYSTEM.

TAILOR
TO CHANGE A SYSTEM OR PROGRAM TO SUIT A SPECIAL NEED OR PURPOSE. (ANSI-X3H1)

TARGET LANGUAGE
THE LANGUAGE IN WHICH A DESIRED PROGRAM IS TO BE EXPRESSED. (DAN 265)

TARGET MACHINE
THE COMPUTER ON WHICH A PARTICULAR COMPUTER PROGRAM IS DESIGNED TO BE USED. (NASA)

TASK MILESTONE
A MAJOR VISIBLE EVENT OR INTERFACE DURING A PROJECT. (DAN LD7)

TECHNOLOGY TRANSFER
REFERS TO SOFTWARE TECHNOLOGY TRANSFER; SEE EDUCATION

TELEMETRY/TRACKING
THIS INCLUDES ALL SOFTWARE COMPONENTS WHICH ARE SPECIFICALLY REQUIRED TO INTERFACE (EITHER READ, WRITE, OR FORMAT) WITH TELEMETRY OR TRACKING DATA. (SEL)

TERMINAL INTERFACE PROCESSOR (TIP)
A PACKET-SWITCH HARDWARE USED FOR STORAGE OF MESSAGES, ROUTING OF SIGNALS, AND COMMUNICATIONS IN THE ARPANET. IT HAS GREATER FLEXIBILITY, AND CAN HANDLE MORE LINES AND HOSTS THAN THE FUNCIONALLY SIMILAR IMP.

TERMINAL SIMULATOR
A COMPUTER PROGRAM USED TO PRESENT INPUT MESSAGES TO THE CONTROL PROGRAM SO AS TO APPEAR THAT THEY HAD BEEN INPUT FROM AN ACTUAL TERMINAL DEVICE. (DAN LD7)

TERMINATION PROOF
PROOF THAT A PROGRAM DOES TERMINATE.

TEST
ANY PROGRAM OR PROCEDURE THAT IS DESIGNED TO OBTAIN, VERIFY, OR PROVIDE DATA FOR THE EVALUATION: RESEARCH AND DEVELOPMENT (OTHER THAN LABORATORY EXPERIMENTS): PROGRESS IN ACCOMPLISHING DEVELOPMENT OBJECTIVES: OR PERFORMANCE AND OPERATIONAL CAPABILITY OF SYSTEMS, SUBSYSTEMS, COMPONENTS, AND EQUIPMENTS ITEMS. (AFR 80-14)

TEST BEDS
A TEST SITE THAT EITHER CONTAINS THE ACTUAL HARDWARE AND INTERFACES (HARDWARE TEST BED) OR SIMULATES THEM (SOFTWARE TEST BED). 1. HARDWARE TEST BED - INCLUDES ACTUAL COMPUTER AND INTERFACE HARDWARE, THUS PERMITTING ACTUAL CHECKOUT OF HARDWARE/SOFTWARE INTERFACES AND ACTUAL INPUT-OUTPUT. THE PROGRAM EXECUTION IS CONFIRMED USING ACTUAL HARDWARE TIMING CHARACTERISTICS, BUT THE OUTPUT IS LIMITED, AND IT HAS LIMITED DIAGNOSTIC CAPABILITIES. 2. SOFTWARE TEST BED - USES AN INSTRUCTION SIMULATOR TO SIMULATE ACTUAL HARDWARE. THE APPROACH POORLY REPRESENTS ACTUAL I/O, RUNS 7 TO 15 TIMES REAL-TIME, AND IS AN EXPENSIVE METHOD OF CONDUCTING LENGTHY TESTING OF SOFTWARE. THE APPROACH PERMITS FULL CONTROL OF INPUTS AND COMPUTER CHARACTERISTICS, ALLOWS PROCESSING OF INTERMEDIATE OUTPUTS WITH OUT DESTROYING REAL TIME, AND ALLOWS FULL TEST REPEATABILITY AND DIAGNOSTICS. (DAN 134)

TEST CONTROL PROGRAM
USUALLY REFERS TO SCENARIO OR THE PROGRAM READING AND INTERPRETING A SCENARIO, BUT COULD BE ANY SOFTWARE WHICH PROVIDES AUTOMATIC DIRECTION FOR TESTING. (DAN 1201)

TEST DATA
TEST ENVIRONMENT DATA THAT ARE PREPARED MANUALLY OR BY AN AUTOMATIC TEST CASE GENERATOR (DAN LD7)

TEST DATA BASE
COLLECTION OF DATA STORED ON A COMPUTER PERIPHERAL DEVICE (E.G., TAPE, DISK, THAT CLOSELY MATCHES THE "REAL" DATA BASE). IDEALLY, A TEST DATA BASE SHOULD BE IDENTICAL TO A REAL DATA BASE BUT USUALLY IT ONLY PROVIDES REPRESENTATIVE DATA. DAN 154)

TEST DATA GENERATION
THE PREPARATION, BY MANUAL OR AUTOMATED MEANS, OF DATA TO BE USED IN TESTING A PROGRAM OR SYSTEM.

TEST DESIGN
THE SELECTION OF THE OPTIMUM METHODS, TOOLS, PROCEDURES, AND TEST CASES TO BE USED IN TESTING A SOFTWARE PROGRAM OR SYSTEM.

TEST DRIVERS
SOFTWARE TEST DRIVERS ARE TOOLS WHICH PROVIDE THE FACILITIES FOR EXECUTING THE TEST HARDWARE/SOFTWARE BY LOADING INPUT DATA FILES WHICH REPRESENT THE TEST SITUATION OR AN EVENT TO YIELD RECORDED DATA IN ORDER TO EVALUATE ACAINST EXPECTED RESULTS. TEST DRIVERS ARE RESTRICTED TO OPERATION IN THE SAME HOST ENVIRONMENT AS THE TEST ARTICLE. INPUT/OUT FUNCTIONS MAY BE BYPASSED OR MODIFIED BY TEST DRIVER SOFTWARE AND ARE GENERALLY DESIGNED TO FACILITATE THE PREPARATION OF INPUT AND THE EVALUATION OF TEST OUTPUT DATA. TEST DRIVERS MAY OPERATE IN EITHER STATIC OR DYNAMIC MODES. STATIC TEST DRIVERS MAY BE EITHER ONE-SHOT OR REPETITIVE. THIS TYPE OF DRIVER USUALLY PROVIDES AT LEAST THE FACILITIES FOR: A. READING TEST INPUT DATA INTO SPECIFIC DATA BASE LOCATIONS. B. PASSING CONTROL TO THE TEST ARTICLE OR ITS ASSOCIATED EXECUTIVE SOFTWARE. C. RESUMING CONTROL AFTER EACH EXECUTION AND READING OUT OR STORING TEST RESULTS. DYNAMIC TEST DRIVERS OPERATE IN REAL TIME OR NEAR-REAL TIME TO PROVIDE A MORE REALISTIC RUN-TIME ENVIRONMENT THAN IS POSSIBLE WITH STATIC DRIVERS. ALL OF THE FACILITIES LISTED FOR STATIC DRIVERS ARE USUALLY PROVIDED BY A DYNAMIC TEST DRIVER. ADDITIONALLY, A TEST EXECUTIVE IS GENERALLY PROVIDED TO SEQUENCE TEST AND DATA TRANSFER OPERATIONS. OTHERWISE, THE OPERATIONAL EXECUTIVE MUST BE DESIGNED TO INTERFACE WITH A DYNAMIC TEST DRIVER. TEST DRIVERS ARE OFTEN DESIGNED TO WORK WITH A VARIETY OF OFF-LINE UTILITY AND SUPPORT COMPUTER PROGRAMS. SOME OF THESE ARE VARIOUSLY KNOWN AS TEST GENERATORS/DISTRIBUTORS, SCENARIO GENERATORS, TEXT EDITORS, TEST REPORT GENERATORS, AND DATA REDUCTION AND ANALYSIS SOFTWARE. (SET) SEE ALSO: ENVIRONMENT SIMULATOR (2) TO RUN TESTS IN A CONTROLLED MANNER, IT IS OFTEN NECESSARY TO WORK WITHIN THE FRAMEWORK OF A "SCENARIO" - A DESCRIPTION OF A DYNAMIC SITUATION TO ACCOMPLISH THIS, IT IS NECESSARY TO LOAD THE INPUT DATA FILES FOR THE SYSTEM WITH DATA VALUES REPRESENTING THE TEST SITUATION OR EVENTS TO YIELD RECORDED DATA TO EVALUATE AGAINST EXPECTED RESULTS. THESE AIDS PERMIT RELATIVELY EASY GENERATION OF DATA IN EXTERNAL FORM TO BE ENTERED AUTOMATICALLY INTO THE SYSTEM AT THE PROPER TIME. (DAN 134) (3) SEE ALSO: DRIVERS, DRIVER PROGRAMS, TEST MODULE DRIVER

TEST GRAMMAR
A TEST GRAMMAR IS A CONTEXT-FREE GRAMMAR WHICH DESCRIBES THOSE ASPECTS OF A PROGRAM TO BE TESTED, AS WELL AS THE ASSUMPTIONS AS TO WHICH TEST CASES ARE CONSIDERED EQUIVALENT. THE GRAMMAR GENERATES TEST DATA IN LEVELS OF EVER INCREASING COMPLEXITY OF TEST CASES. AT EACH LEVEL THE PROGRAMMER MAY USE THE RESULTS OF TESTING AT PREVIOUS LEVELS TO STRENGTHEN THE ASSUMPTIONS ON THE TEST GRAMMAR, THEREBY, REDUCING THE NUMBER OF TEST CASES GENERATED AT

SUBSEQUENT LEVELS. (DAN 837)

TEST MANAGEMENT
MANAGEMENT PROCEDURES DESIGNED TO CONTROL IN AN ORDERED WAY A LARGE AND EVOLVING AMOUNT OF PIECES OF INFORMATION ON SYSTEM FEATURES TO BE TESTED, ON SYSTEM IMPLEMENTATION PLANS, AND ON TEST RESULTS. (DAN 529)

TEST METHODOLOGIES
THE PROCEDURES AND TOOLS UTILIZED IN PROVING THAT A PROGRAM CORRECTLY FULFILLS ITS REQUIREMENTS. (DAN 1201)

TEST MODULE DRIVER
INDEPENDENT MODULES WHICH EXECUTE UNDER THE SAME OPERATING SYSTEM AS THE SOFTWARE TO BE TESTED AND PERFORM SPECIFIC TESTS IN RESPONSE TO EXTERNAL STIMULI. (DAN 1201) SEE ALSO DRIVERS, DRIVER PROGRAMS

TEST PLAN
A MANAGEMENT DOCUMENT THAT DESCRIBES HOW AND WHEN SPECIFIED TEST OBJECTIVES WILL BE MET. (DAN 134) (2) THE TEST PLAN USUALLY CONTAINS THE FOLLOWING INFORMATION: A) A GENERAL TEST PHILOSOPHY OR STRATEGY. B) A FUNCTIONAL DESCRIPTION OF WHAT IS BEING TESTED. C) A REPRESENTATION OF FUNCTIONAL COVERAGE (I.E. MATRIX, CAUSE AND EFFECT, FAMILY TREE, ETC.) D) A DESCRIPTION OF WHAT EACH TEST CASE WILL TEST AND HOW IT WILL BE ACCOMPLISHED. 3) TESTING DEPENDENCIES (BUILD REQUIREMENT, HARDWARE/SIMULATOR NEEDS, ETC.) F) ENTRANCE AND EXIT CRITERIA. (DAN 781) (3) A FORMAL DOCUMENT WHICH DEFINES THE TESTS TO BE PERFORMED TO VERIFY THAT THE COMPUTER PROGRAM MEETS THE PERFORMANCE REQUIREMENTS STATED AS THE ACCEPTANCE CRITERIA DEFINED IN THE PROGRAM PERFORMANCE SPECIFICATION. (DAN 1201)

TEST PLAN DOCUMENT
A PLAN THAT TESTS THE EFFECTIVENESS OF THE OBJECT SYSTEM, AS IMPLEMENTED, AND DETERMINES HOW IT CAN BE VALIDATED, VERIFIED, AND CERTIFIED. THIS DOCUMENT IS INPUT TO THE SUBSEQUENT FUNCTIONS OF THE QUALITY ASSURANCE PROCESS AND TO THE PROGRAM VALIDATION FUNCTION. (DAN LD7)

TEST PLAN INSPECTION
A TEST PLAN INSPECTION IS A FORMAL METHOD OF EXAMINING THE TEST PLAN. THE PURPOSE OF THE INSPECTION IS TO ASSURE THE COMPLETENESS AND ACCURACY OF THE TEST PLAN. (DAN 781)

TEST PROCEDURE
A FORMAL DOCUMENT DEVELOPED FROM A TEST PLAN THAT PRESENTS DETAILED INSTRUCTIONS FOR THE SET UP, OPERATION, AND EVALUATION RESULTS FOR EACH DEFINED TEST. (DAN 1201)

TEST REPORT
A DOCUMENT RECORDING THE RESULTS OF A PROGRAM TEST. (DAN 1201)

TEST RESULT PROCESSOR
A COMPUTER PROGRAM USED TO PERFORM TEST OUTPUT DATA REDUCTION. FORMATTING AND PRINTING. SOME PERFORM STATISTICAL ANALYSIS WHERE THE ORIGINAL DATA MAY BE THE OUTPUT OF A MONITOR. (DAN 134)

TEST SYSTEM

THE SYSTEM OF HARDWARE, OPERATIONAL SOFTWARE AND TEST SOFTWARE INTEGRATED AND USED TO RUN PRODUCT ACCEPTANCE TESTS ON THE OPERATIONAL SOFTWARE. (DAN 1201)

TEST TOOLS
THE SUPPORT SOFTWARE USED AS AN AID IN PROGRAM CHECKOUT AND DEBUGGING. (DAN 1201) SEE ALSO: AUTOMATED TESTING, SUPPORT SOFTWARE, AUTOMATED VERIFICATION TOOLS

TEST VALIDITY
THE DEGREE TO WHICH A TEST ACCOMPLISHES ITS SPECIFIED GOAL. (DAN 1201) SEE ALSO: TESTEDNESS

TESTABILITY
CODE POSSESSES THE CHARACTERISTIC TESTABILITY TO THE EXTENT THAT IT FACILITATES THE ESTABLISHMENT OF VERIFICATION CRITERIA AND SUPPORTS EVALUATION OF ITS PERFORMANCE. THIS IMPLIES THAT REQUIREMENTS ARE MATCHED TO SPECIFIC MODULES, OR DIAGNOSTIC CAPABILITIES ARE PROVIDED, ETC. (DAN 239)

TESTABLE
A SOFTWARE PRODUCT IS TESTABLE TO THE EXTENT THAT IT FACILITATES THE ESTABLISHMENT OF VERIFICATION CRITERIA AND SUPPORTS EVALUATION OF ITS PERFORMANCE...SOME OF THE CHARACTERISTICS WHICH INCREASE TESTABILITY ARE: (A) FUNCTIONAL MODULARITY, WHICH FACILITATES THE MATCHING OF REQUIREMENTS TO SPECIFIC MODULES OF A PROGRAM. (B) CAPABILITY OF PROVIDING DIAGNOSTICS, E.G. THROUGH USE OF CONDITIONAL ASSEMBLY TO INVOKE MACRO GENERATION OF CODE FOR DIAGNOSTIC PRINTOUTS, UNDER USER CONTROL. (C) AUXILIARY CODE IS USED TO EVALUATE CERTAIN INVARIANTS (E.G., CODE IS ADDED TO CALCULATE THE TOTAL ENERGY FOR VARIOUS STATES OF A CONSTANT ENERGY SYSTEM). (D) COMMENTS INDICATING UNACCEPTABLE VALUES AND THE RECOMMENDED DEFAULT ACTION ARE PLACED AT A POINT WHERE INTERMEDIATE OR REQUIRED OUTPUT IS DEFINED. (SET)

TESTEDNESS
TESTEDNESS IS A DYNAMIC MEASURE WHICH INDICATES THE EXTENT TO WHICH A PARTICULAR PIECE OF SOFTWARE HAS BEEN TESTED BY PARTICULAR TEST CASES. THE MEASURE IS DEFINED IN SUCH A WAY THAT WHEN MEASURING THE EXTENT TO WHICH A LOGICAL STRUCTURE OR PART OF A LOGICAL STRUCTURE HAS BEEN EXERCISED OR TESTED; THE TESTEDNESS OF A NODE INCREASES AS THE NUMBER OF TIMES IT IS EXERCISED INCREASES AND DECREASES AS THE PROBABILITY OF ERROR OR THE ACCESSIBILITY INCREASES. (DAN 766)

TESTING
TESTING IS THE PART OF THE SOFTWARE DEVELOPMENT PROCESS WHERE THE COMPUTER PROGRAM IS SUBJECT TO SPECIFIC CONDITIONS TO SHOW THAT THE PROGRAM MEETS ITS INTENDED DESIGN. IT IS THE PROCESS OF FEEDING SAMPLE INPUT DATA INTO A PROGRAM, EXECUTING IT, AND INSPECTING THE OUTPUT AND/OR BEHAVIOR FOR CORRECTNESS. THE CORNERSTONE OF RELIABILITY METHODOLOGY IS TESTING. TRADITIONALLY, TESTING IS THE DEVELOPMENT PHASE WHERE THE LARGEST QUANTITY OF ERRORS IS DETECTED AND CORRECTED. BUT, GIVEN THIS EXPENDITURE, THE SOFTWARE DEVELOPER HAS NO REAL ASSURANCE OF DEVELOPING ERROR-FREE SOFTWARE, FOR THE TESTING CYCLE ONLY DEMONSTRATES THE PRESENCE OF ERROR. THE FOLLOWING TECHNIQUES OR TOOLS ARE CONSIDERED PART OF THE TESTING CYCLE: ANALYZERS, ASSERTIONS, SOURCE LANGUAGE DEBUG, INTENTIONAL FAILURE, TEST DRIVERS, REGRESSION TESTING, ENVIRONMENT SIMULATORS, STANDARDIZED TESTING, SYMBOLIC

EXECUTION , INTERACTIVE DEBUG, FORLIGN DEBUG, SNEAK CIRCUIT ANALYSIS... ALSO SEE ANALYZERS, ASSERTIONS, SOURCE LANGUAGE DEBUG, TEST DRIVERS, REGRESSION TESTING, ENVIRONMENT SIMULATORS, INTERACTIVE DEBUG, AND FOREIGN DEBUG. (SET) (2) EXERCISING DIFFERENT MODES OF COMPUTER PROGRAM OPERATION THROUGH DIFFERENT COMBINATIONS OF INPUT DATA (TEST CASES) TO FIND ERRORS. (IEEE TASK GROUP FOR STANDARDIZATION OF SOFTWARE TEST DOCUMENTATION)

TESTING CRITERIA
THE REQUIREMENTS THE PROGRAM UNDER TEST MUST SATISFY. (DAN 1201)

TESTING EFFECTIVENESS
THE DEGREE TO WHICH THE SOFTWARE CAN BE CHECKED OUT WITH ALL "BUGS" REMOVED. (DAN 1201)

TESTING OBJECTIVE
A GOAL TO BE ATTAINED FROM TESTING SOFTWARE. (DAN 1201)

TESTING PHASE
DURING THE TEST PHASE, SYSTEM INTEGRATION OF THE SOFTWARE COMPONENTS AND SYSTEM ACCEPTANCE TESTS ARE PERFORMED AGAINST THE REQUIREMENTS. (SET) (2) THE DESIGN OF TESTS, TESTING STRATEGIES, AND THE RUNNING OF SUCH TESTS. (SEL)

TESTMASTER
AN AUTOMATIC SOFTWARE TEST DRIVER AVAILABLE FROM HOSKYNS, INC. ORIENTED TOWARD TESTING OF COBOL PROGRAMS RUNNING ON AN IBM 360/370 SYSTEM. (DAN286)

TEXT DATA
PROGRAM DOCUMENATION THAT IS PREPARED MANUALLY AND UPDATED BY A TEXT EDITOR. (DAN ED7)

TEXT-FORMATTING APPLICATIONS
INDEXING TERM. REFERS TO SOFTWARE USED AS A COMPONENT IN A TEXT-FORMATTING SYSTEM, OR TO THE DEVELOPMENT OF THE SOFTWARE FOR A TEXT-FORMATTING SYSTEM.

TEXT-PROCESSING APPLICATIONS
INDEXING TERM. REFERS TO SOFTWARE USED AS A COMPONENT IN A TEXT-PROCESSING SYSTEM, OR TO THE DEVELOPMENT OF THE SOFTWARE FOR A TEXT-PROCESSING SYSTEM.

THEOREM PROVER
A SOFTWARE PACKAGE THAT AUTOMATICALLY OR SEMI-AUTOMATICALLY EVALUATES INPUTTED THEOREMS. IT IS USED TYPICALLY AS THE FINAL STAGE OF A PROOF OF CORRECTNESS SYSTEM.

THROUGHPUT
A MEAURE OF THE AMOUNT OF WORK PERFORMED BY A COMPUTING SYSTEM OVER A GIVEN PERIOD OF TIME E.G., JOBS PER DAY. (2) A MEASURE OF COMPUTER CAPACITY IN TERMS OF EXECUTION RATE AND WORD SIZE. (NASA)

TIER CHART
A TREE-GRAPH REPRESENTATION OF A PROGRAM AND ITS NAMED MODULES, IN WHICH THE SUBORDINATION RELATION IS INVOCATION. SUBROUTINE INVOCATION NODES OCCUR MORE THAN ONCE; HOWEVER, ALL BUT ONE OF THESE NODES APPEAR AS LEAVES OF THE TREE, AND THE OTHER FORMS THE ROOT OF THE SUBROUTINE TIER HIERARCHY. (DAN 1153)

TIME DOMAIN
AN APPROACH TO SOFTWARE RELIABILITY ESTIMATION WHICH PREDICTS SOFTWARE FAILURES AS A FUNCTION OF TIME. REASONABLE ESTIMATES OF MEAN TIME TO FAILURE AND THE CHECKOUT TIME REQUIRED TO ACHIEVE A GIVEN LEVEL OF ASSURANCE FOR PROGRAM CORRECTNESS CAN BE CALCULATED BY FITTING OBSERVED DATA TO A POSTULATED FAILURE TIME DISTRIBUTION, OR BY CALCULATING THE MOMENTS OF THE DISTRIBUTION (MEAN, STANDARD DEVIATION) AND MATCHING THE ACTUAL TO THE THEORETICAL IN THE SENSE OF MAXIMUM LIKELIHOOD ESTIMATORS.

TIMESHARING
A MODE OF OPERATION THAT PROVIDES FOR THE INTERLEAVING OF TWO OR MORE INDEPENDENT PROCESSES ON ONE FUNCTIONAL UNIT. (ANSI-X3)

TIMING ANALYZER
A COMPUTER PROGRAM THAT MONITORS AND PRINTS EXECUTION TIME OF ALL PROGRAMS ELEMENTS (FUNCTIONS, ROUTINES, AND SUBROUTINES). (DAN 134)

TIP
SEE: TERMINAL INTERFACE PROCESSOR

TOLERANCE
A SYSTEM'S INPUT DATA TOLERANCE IS A MEASURE OF THE SYSTEM'S ABILITY TO ACCEPT DIFFERENT FORMS OF THE SAME INFORMATION AS VALID, (I.E. WITHOUT MALFUNCTION OR REJECTION) (DAN 781) (2) NEARLY SYNONOMOUS WITH ROBUSTNESS.

TOP DOWN
IN DESIGNING COMPUTER PROGRAMS, THE TOP-DOWN APPROACH IDENTIFIES MAJOR FUNCTIONS TO BE ACCOMPLISHED, THEN PROCEEDS FROM THERE TO AN IDENTIFICATION OF THE LESSER FUNCTIONS THAT DERIVE FROM THE MAJOR ONES. THE DEFINITION OF "TOP-DOWN" IS THE MIRROR IMAGE OF "BOTTOM-UP" WHERE THE LOWER PROCEDURES ARE WRITTEN FIRST, AND UPPER LEVELS LATER. TOP-DOWN DESIGN INVOLVES BREAKING A LARGE PROGRAM INTO SMALLER SUBPROGRAMS THAT CAN BE DEALT WITH INDIVIDUALLY. TOP-DOWN CONCEPTS CAN BE APPLIED TO CODING AND TESTING. (SET) SEE ALSO: HIERARCHICAL STRUCTURE, LEVEL OF ABSTRACTION, STEP-WISE REFINEMENT, STRUCTURED PROGRAMMING. (2) THE DESIGN (OR IMPLEMENTATION) OF THE SYSTEM, STARTING WITH A SINGLE COMPONENT, ONE LEVEL AT A TIME, BY EXPANDING EACH COMPONENT REFERENCE AS AN ALGORITHM POSSIBLY CALLING OTHER NEW COMPONENTS. (SEL) (3) A GENERAL TERM INDICATING A HEIRARCHY OF DEPENDENT ELEMENTS AND AN ORDER OF ANALYSIS, DEFINITION, DESIGN OR PRODUCTION BEGINNING WITH THE MOST COMMON ELEMENT(S) (TOP) TO THE LEAST COMMON ELEMENTS (ON DOWN). (DAN 1201)

TOP DOWN DEVELOPMENT
TECHNIQUE FOR IMPLEMENTING HIERARCHICALLY STRUCTURED PROGRAMS. HERE THE TOP-LEVEL ROUTINES ARE WRITTEN FIRST AND LOWER LEVEL ROUTINES, CALLED STUBS, ARE WRITTEN TO INTERFACE WITH THESE. (DAN 237) ONCE REQUIREMENTS ARE FIRMED UP, THE DEVELOPMENT PROCESS DECOMPOSES THE PROPOSED SYSTEM INTO A SERIES OF LEVELS IN A HIERARCHY, BEGINNING AT THE TOP AND WORKING DOWN. THE HIGHEST LEVEL IS THEN DESIGNED, CODED,AND SUBSEQUENTLY TESTED FIRST, USING STUBS WITH DUMMY CODE TO STAND IN FOR LOWER-LEVEL UNITS THAT ARE INVOLVED, AND SO ON. (DAN 227)

TOP-DOWN DESIGN
TOP DOWN DESIGN IMPLIES AN ORDERING TO THE SEQUENCE OF DECISIONS WHICH ARE MADE IN THE DECOMPOSITION OF A SOFTWARE SYSTEM, BY BEGINNING WITH A SIMPLE

DESCRIPTION OF THE ENTIRE PROCESS (TOP LEVEL). THROUGH A SUCCESSION OF REFINEMENTS OF WHAT HAS BEEN DEFINED AT EACH LEVEL, LOWER LEVELS ARE SPECIFIED. (SET) (2) A DESIGN AND CODING CRITERION IN WHICH OPERATIONS ARE DEFINED IN A HIERARCHICAL MANNER AND FROM THE MORE GENERAL TO THE MORE PRECISE. TOP LEVEL OPERATIONS ARE DEFINED IN RELATIVELY GENERAL TERMS. OPERATIONS ON ALL LEVELS (EXCEPT THE BOTTOM ONE) ARE DEFINED MORE PRECISELY IN TERMS OF OPERATIONS ON THE LEVEL IMMEDIATELY BELOW. THE DEFINING OF A MORE GENERAL OPERATION IN TERMS OF MORE SPECIFIC, LOWER LEVEL OPERATIONS IS CALLED STEPWISE REFINEMENT. (ABBOTT)

TOP-DOWN IMPLEMENTATION
A DEVELOPMENTAL METHODOLOGY WHOSE DISTINGUISHING FEATURE IS THAT HIGHER LEVELS OF THE PROGRAM ARE IMPLEMENTED (DESIGNED, CODED, TESTED) BEFORE THE LOWER LEVELS ARE IMPLEMENTED. THIS METHODOLOGY IMPLIES THAT SYSTEM INTEGRATION PROCEEDS FROM THE HIGHEST LEVEL TO THE LOWEST LEVEL BY INTEGRATING SUCCESSIVELY LOWER-LEVEL MODULES/COMPONENTS INTO SUCCESSFULLY INTEGRATED HIGHER-LEVEL MODULES/COMPONENTS.

TOP-DOWN PARSING
TOP-DOWN DECOMPOSITION BY DIJKSTRA: A PARSING IS A DECOMPOSITION

TOP-DOWN PROGRAM DEVELOPMENT
DOWNWARD FROM THE TOP LEVEL OF PROGRAM DESIGN TO THE BOTTOM LEVELS, CONTINUOUSLY EXERCISING THE ACTUAL INTERFACES BETWEEN PROGRAM MODULES. THE BOTTOM LEVEL MAY BE STANDARD PACKAGED ROUTINES - DATA ACCESS METHODS, SORT ROUTINES OR INTERFACES WITH OTHER SYSTEMS. THIS APPROACH IS THE OPPOSITE OF THE USUAL ONE OF CHECKING THE BOTTOM-LEVEL MODULES FIRST AND WORKING UP, FINALLY INTEGRATING AND TESTING THE ENTIRE SYSTEM. HERE, THE INTEGRATION OF MODULES IS A CONTINUOUS PROCESS. (DAN LD7)

TOP-DOWN PROGRAMMING
THE CONCEPT OF PERFORMING IN HIERARCHICAL SEQUENCE A DETAILED DESIGN, CODE, INTEGRATION AND TEST AS CONCURRENT OPERATIONS. (DAN 140)

TOP-DOWN PROGRAMMING PROCESS
AN EXPANSION OF FUNCTIONAL SPECIFICATIONS TO SIMPLER AND SIMPLER FUNCTIONS UNTIL, FINALLY, STATEMENTS OF THE PROGRAMMING LANGUAGE ITSELF ARE REACHED. (SET)

TOP-DOWN STRUCTURED PROGRAM
(TDSP) A STRUCTURED PROGRAM WITH THE ADDITIONAL CHARACTERISTICS OF THE SOURCE CODE BEING LOGICALLY, BUT NOT NECESSARILY PHYSICALLY, SEGMENTED IN A HIERARCHICAL MANNER AND ONLY DEPENDENT ON CODE ALREADY WRITTEN. CONTROL OF EXECUTION BETWEEN SEGMENTS IS RESTRICTED TO TRANSFERS BETWEEN ADJACENT HIERARCHICAL SEGMENTS. (DAN 140) (2) A PROGRAM WHICH SATISFIES THE DESIGN CRITERIA OF TOP DOWN DESIGN AND STRUCTURED CODE. (ABBOTT)

TOP-DOWN STRUCTURED PROGRAMMING
THE PROCESS OF DEVELOPING TOP DOWN STRUCTURED PROGRAMS. ASSOCIATED WITH TOP DOWN STRUCTURED PROGRAMMING ARE CERTAIN PRACTICES SUCH AS INDENTATIONS OF SOURCE CODE TO REPRESENT LOGIC LEVELS, THE USE OF INTELLIGENT DATA NAMES AND DESCRIPTIVE COMMENTARY. TOP DOWN STRUCTURED PROGRAMMING REQUIRES TOP DOWN PROGRAMMING AS THE PRIMARY IMPLEMENTATION METHODOLGY. (DAN 140)

TOP-DOWN TESTING
IF MODULES ARE PRODUCED IN A TOP DOWN ORDER, THEN TOP DOWN TESTING CAN ALSO BE EMPLOYED USING A PARTIALLY COMPLETED SYSTEM IN WHICH LOWER LEVELS ARE REPRESENTED BY PROGRAM "STUBS" AND PROGRAMS ARE EXECUTED IN THE ENVIRONMENT IN WHICH THEY WILL ACTUALLY OPERATE. THIS APPROACH ALLOWS FOR EARLIER INTEGRATION AND TESTING WHICH SHOULD UNCOVER PROBLEMS SOONER THEN CONVENTIONAL TESTING WHERE INTEGRATION IS THE LAST STEP. (SET)

TOTAL CORRECTNESS
A PROGRAM IS TOTALLY CORRECT WITH RESPECT TO AN INTENDED FUNCTION IF IT (1) IS IS PARTIALLY CORRECT WITH RESPECT TO THAT FUNCTION, AND (2) TERMINATES AFTER A FINITE LENGTH OF TIME ON EACH INPUT FOR WHICH THE FUNCTION IS DEFINED. MOST METHODS FOR PROVING TOTAL CORRECTNESS REQUIRE A SEPARATE PROOF FOR TERMINATION, USUALLY BASED ON A WELL-ORDERING OF PROGRAM STATES. (SET) (2) IF A PROGRAM BOTH TERMINATES AND SATISFIES ITS OUTPUT SPECIFICATION, THAT PROGRAM IS SAID TO BE TOTALLY CORRECT. (DAN 419)

TRACE
A RECORD OF PROGRAM EXECUTION SHOWING THE SEQUENCE OF SUBROUTINE AND FUNCTION CALLS, AND SOMETIMES THE VALUE OF SELECTED VARIABLES. CODE USED IN PRODUCING A TRACE IS AUTOMATICALLY INSERTED INTO A PROGRAM, USUALLY BY THE COMPILER, SOMETIMES BY OTHER SUPPORT SOFTWARE. (SEL) SEE ALSO: SOURCE LANGUAGE DEBUG.

TRACE PROGRAM
A COMPUTER PROGRAM THAT RECORDS THE CHRONOLOGICAL SEQUENCE OF EVENTS TAKEN BY A TARGET PROGRAM DURING ITS EXECUTION. (DAN 134)

TRACER PROGRAM
A PROGRAM ANALYSIS TOOL WHICH WILL ANALYZE COMPUTER PROGRAMS LOOKING FOR "DEAD CODE" - I.E., CODE IN A PROGRAM WHICH CAN NOT BE EXECUTED. (DAN 142)

TRANSFORMATION (OF DATE)
THE PROCESS OF CHANGING DATA FROM ONE FORM TO ANOTHER FORM. TRANSFORMATION WORK IS THE MEASURE OF THE ENERGY OR RESOURCES NEEDED TO CONVERT DATA FROM SOME ORIGINAL STATE TO ITS TRANSFORMED STATE. (DAN 781) (2) TRANSFORMATION WORK CAN BE AN INPUT TO PERFORMANCE EVALUATION.

TRANSLATION
THE CONVERSION OF A COMPUTER PROGRAM FROM ONE PROGRAMMING LANGUAGE TO AN EQUIVALENT PROGRAM IN A DIFFERENT LANGUAGE, FREQUENTLY IN REFERENCE TO COMPILATION OR ASSEMBLY. (NASA)

TRANSLATOR
A COMPUTER PROGRAM WRITTEN TO ACCEPT INFORMATION FROM ONE SYSTEM OF REPRESENTATION AND CONVERT THIS INTO EQUIVALENT INFORMATION IN ANOTHER SYSTEM OF REPRESENTATION. (DAN 134)

TRANSPORTABILITY
THE CAPABILITY OF A PROGRAM TO OPERATE ON VARIOUS DIFFERENT MAKE AND MODEL COMPUTERS WITH A MINIMUM OF MODIFICATION REQUIRED. (DAN 1201)

TRAP
A TECHNIQUE IN WHICH THE LOGIC FLOW OF A PROGRAM IS INTERRUPTED FOR THE

PURPOSE OF SETTING ASIDE INTERIM RESULTS FOR TEST MEASUREMENT OR PERFORMING SPECIAL TEST FUNCTIONS. (DAN 134)

TREE
AN ACYCLIC CONNECTED GRAPH. IF THE TREE HAS N >OR= NODES, THEN IT ALSO HAS N-1 EDGES. EVERY PAIR OF NODES IS CONNECTED BY EXACTLY ONE PATH. THE TREE OFTEN REPRESENTS A HEIRARCHY, IN WHICH EDGES ARE DIRECTED TO DENOTE A SUBORDINATING RELATIONSHIP BETWEEN THE TWO JOINED NODES. (DAN 1153)

TRI-SERVICE
ARMY, NAVY, AIR FORCE- ADJECTIVE APPLIED TO JOINT EFFORTS OF THE 3 SERVICES. (DAN 222)

TRUSTED SOFTWARE
SOFTWARE USED FOR MULTI-LEVEL OPERATING SYSTEMS THAT ENSURES, TO THE BEST TECHNOLOGY CURRENTLY AVAILABLE, A REASONABLE TRUST IN COMPLETE SEPERATION AND NON-INTERFERENCE IN THE MULTI-LEVELS. SOMETIMES USED SYNONOMOUSLY WITH "SECURE SOFTWARE".

TYPE EQUIVALENCE
A TERM USED TO DENOTE DATA STRUCTURES, FUNCTIONS, OR MODULES WHICH MAY BE USED BY MORE THAN ONE PROGRAM OR SUBPROGRAM. FOR THE PURPOSE OF THE LINKAGE EDITOR ROUTINE, TWO TYPES ARE EQUIVALENT ONLY IF THEY ARE IDENTICAL, INCLUDING HAVING THE SAME NAME (OR IF THEIR NAMES ARE ALIASES). ALSO, TWO TYPES ARE EQUIVALENT IF THEY ARE STRUCTURALLY ISOMORPHIC. (DAN 243)

TYPE (DATA)
SEE DATA TYPE

TYPE OF SOFTWARE
THE FOUR MAJOR CLASSIFICATIONS OF MOST OF THE APPLICABLE SOFTWARE BEING DEVELOPED ARE : SCIENTIFIC, BUSINESS/FINANCIAL, SYSTEMS AND UTILITY. THESE CLASSIFICATIONS MAY BE REFINED INTO THE CATEGORIES OF: STRING PROCESSING, DATA BASE APPLICATIONS, REAL TIME, AND TABLE HANDLER. A FURTHER REFINEMENT INCLUDES THE CATEGORIES OF: ATTITUDE/ORBIT, TELEMETRY/TRACKING, COMMAND/CONTROL, MATHEMATICAL, AND NUMERICAL ON-BOARD. (SEL)

UNDERSTANDABLE
A SOFTWARE PRODUCT IS UNDERSTANDABLE TO THE EXTENT THAT ITS PURPOSE IS CLEAR TO THE INSPECTOR...MANY TECHNIQUES HAVE BEEN PROPOSED TO INCREASE UNDERSTANDABILITY. PROMINENT AMONG THESE ARE CODE STRUCTUREDNESS WHICH SIMPLIFIES LOGICAL FLOW, LOCAL COMMENTARY, TO EXPLAIN COMPLEX CODED INSTRUCTIONS, AND CONSISTENTLY USED MNEMONICS. IN ADDITION, REFERENCES TO READILY AVAILABLE AND UP-TO-DATE DOCUMENTS NEED TO BE INCLUDED IN SOURCE COMMENTARY SO THAT THE INSPECTOR MAY COMPREHEND MORE ESOTERIC CONTENTS. INPUT, OUTPUTS, AND ASSUMPTIONS SHOULD BE STATED IN THE FORM OF GLOSSARIES OR PROSE COMMENTARY. IN GENERAL, A CODING STANDARD ENCOMPASSING FORMAT OF HEADERS AND INDENTATION SHOULD BE FOLLOWED FOR ALL MODULES SO THAT INFORMATION CAN BE FOUND WHERE EXPECTED...ALSO SEE - MAINTAINABLE, STRUCTURED PROGRAMMING, MODULARITY. (SET)

UNIT
A SET OF COMPUTER PROGRAM STATEMENTS TREATED LOGICALLY AS A WHOLE. THE WORD "UNIT" IS RESTRICTED IN THE CONTEXT OF A COMPUTER PROGRAM STRUCTURE. THIS

USAGE DOES NOT REFER TO A DEVICE UNIT, OR LOGICAL UNIT. (SET) (2) A NAMED SUBDIVISION OF A PROGRAM WHICH IS CAPABLE OF BEING STORED IN A PROGRAM SUPPORT LIBRARY AND MANIPULATED AS A SINGLE ENTITY. (DAN 137) SEE ALSO: PROGRAM SEGMENT.

UNIT CONSISTENCY ANALYSIS PROGRAM
A COMPUTER PROGRAM THAT ANALYZES SOURCE CODE TO VERIFY UNITS CONSISTENCY FOR EACH USAGE OF EACH PARAMETER. (DAN 134)

UNIX
AN OPERATING SYSTEM FOR UNIVAC SYSTEMS DEVELOPED BY BELL LABS., PISCATAWAY, N.J.

USER
THE INDIVIDUAL AT THE MAN/MACHINE INTERFACE WHO IS APPLYING THE SOFTWARE TO THE SOLUTION OF A PROBLEM, E.G. TEST OR OPERATIONS. (DAN21) (2) ANY ENTITY USING THE FACILITIES OF AN OPERATING SYSTEM. IN ADDITION TO 'NORMAL' USERS, THIS INCLUDES AT LEAST PROGRAMS, NETWORKS, AND OPERATORS. (ANSI-X3H1)

USER-INTERACTIVE SYSTEM
A COMPUTER SYSTEM WHICH IS USED BY MEANS OF AN INTERACTIVE TERMINAL OPERATED BY THE USER.

UTILITIES
COMPUTER PROGRAMS EMPLOYED BY OTHER V/V/C AIDS TO PROVIDE SPECIAL SERVICES. THESE SERVICES INCLUDE PREPARING PROGRAM DECK LISTINGS, CREATING LOAD TAPES, AND PLOTTING OUTPUT RESULTS. (DAN 134) (2) TOOLS THAT FACILITATE THE PRODUCTION AND CONTROL OF SOFTWARE SYSTEMS. THESE INCLUDE TOOLS FOR DATA SET MANAGEMENT, PROGRAM DEVELOPMENT, AND PROGRAM EXECUTION. (DAN 187) (3) ANY COMPONENT THAT IS GENERATED FOR THE PURPOSE OF SATISYING SOME GENERAL SUPPORT FUNCTION REQUIRED BY OTHER APPLICATIONS SOFTWARE MAY BE CONSIDERED A UTILITY. WE THINK OF THIS CLASS OF COMPONENTS AS CONTAINING SOFTWARE THAT DOES NOT FIT INTO ANY OF THE OTHER THREE CATEGORIES. ALTHOUGH COMPONENTS CAN FALL INTO TWO OF THE PRIMARY CATEGORIES (E.G. SCIENTIFIC AND UTILITY), IT WILL BE EASIER TO USE JUST THE MORE DESCRIPTIVE OF THE CATEGORIES (E.G., VECTOR CROSS PRODUCT-SCIENTIFIC DATA UNPACKING - UTILITY.) (SEL) SEE ALSO: SUPPORT SOFTWARE.

UTILITY SOFTWARE
COMPUTER PROGRAMS WHICH PERFORM VARIOUS SERVICE FUNCTIONS, SUCH AS MOVING PROGRAM AND DATA FILES, AND ACCESSING PERIPHERAL EQUIPMENT (NASA)

VALIDATION
THE PROCESS OF DETERMINING WHETHER EXECUTING THE SYSTEM (I.E., SOFTWARE, HARDWARE, USER PROCEDURES, PERSONNEL) IN A USER ENVIRONMENT CAUSES ANY OPERATIONAL DIFFICULTIES. THE PROCESS INCLUDES ENSURING THAT SPECIFIC PROGRAM FUNCTIONS MEET THEIR REQUIREMENTS AND SPECIFICATIONS. VALIDATION ALSO INCLUDES THE PREVENTION, DETECTION, DIAGNOSIS, RECOVERY, AND CORRECTION OF ERRORS. (2) VALIDATION IS MORE DIFFICULT THAN THE VERIFICATION PROCESS SINCE IT INVOLVES QUESTIONS OF THE COMPLETENESS OF THE SPECIFICATION AND ENVIRONMENT INFORMATION. THERE ARE BOTH MANUAL AND COMPUTER BASED VALIDATION TECHNIQUES. (DAN 154) (3) THE PROCESS OF ENSURING THAT SPECIFIC PROGRAM FUNCTIONS MEET THEIR DETAILED DESIGN REQUIREMENT SPECIFICATIONS. ALSO, SEE PROGRAM VALIDATION, PROGRAM VALIDATION TOOLS, AND VALIDATION AND DEBUGGING

TOOLS. (DAN LD7) SEE ALSO: COMPUTER PROGRAM VALIDATION

VALIDATION AND DEBUGGING TOOLS
TOOLS RELATED TO THE PRODUCTION OF CORRECT, SERVICEABLE PROGRAMS. VALIDATION INCLUDES THE PREVENTION, DETECTION, DIAGNOSIS, RECOVERY, AND CORRECTION OF ERRORS. DEBUGGING INCLUDES CORRECTING ERRORS OF BOTH A LOGICAL AND A CLERICAL NATURE. (DAN LD7)

VALIDATION CRITERIA
A GUIDE DEFINING WHAT WILL BE USED TO DETERMINE COMPLETION OF A MILESTONE. THIS INCLUDES SUCCESSFUL OPERATION OF A TEST TOOL, OR ACCEPTANCE OF A DOCUMENT, OR APPROVAL BY THE REVIEW BOARD. (DAN LD7)

VALIDATION TOOLS
SEE VALIDATION AND DEBUGGING TOOLS

VALUE OF DATA
THE NUMBER AND KIND OF NUMBER (E.G., INTEGER, FLOATING POINT, ASCII ENCODED CHARACTER, ETC.), STORED IN A LOCAL VARIABLE OR DATA AREA, PARAMETER, COMMON VARIABLE, SYSTEM-WIDE DATA ITEM, ETC. (SEL)

VECTOR
ONE DIMENSIONAL ARRAY

VECTOR OPERATION
A UNIQUE OPERATION THAT FOURTH GENERATION HARDWARE CAN PERFORM ON A SET OF DATA THROUGH PARALLEL PROCESSING. IT ALLOWS MULTIPLE 'SCALAR OPERATIONS' TO BE PERFORMED SIMULTANEOUSLY.

VERACITY
VERACITY IS DEFINED AS THE ADEQUACY WITH WHICH A GIVEN ALGORITHM REPRESENTS THE REQUIREMENTS OF THE PHYSICAL WORLD. (DAN 781)

VERIFICATION
COMPUTER PROGRAM VERIFICATION IS THE ITERATIVE PROCESS OF DETERMINING WHETHER OR NOT THE PRODUCT OF EACH STEP OF THE COMPUTER PROGRAM ACQUISITION PROCESS FULFILLS ALL REQUIREMENTS LEVIED BY THE PREVIOUS STEP. THESE STEPS ARE SYSTEM SPECIFICATION VERIFICATION, REQUIREMENTS VERIFICATION, SPECIFICATION VERIFICATION, AND CODE VERIFICATION. (SET) (2) THE PROCESS OF DETERMINING WHETHER THE RESULTS OF EXECUTING THE SOFTWARE PRODUCT IN A TEST ENVIRONMENT AGREE WITH THE SPECIFICATIONS. VERIFICATION IS USUALLY ONLY CONCERNED WITH THE SOFTWARE'S LOGICAL CORRECTNESS (I.E., SATISFYING THE FUNCTIONAL REQUIREMENTS) AND MAY BE A MANUAL OR A COMPUTER BASED PROCESS (I.E., TESTING SOFTWARE BY EXECUTING IT ON A COMPUTER). (DAN 154) (3) THE PROCESS OF ENSURING THAT THE SYSTEM AND ITS STRUCTURE MEET THE FUNCTIONAL REQUIREMENTS OF THE BASELINE SPECIFICATION DOCUMENT. (DAN LD7) ALSO SEE SYSTEM VERIFICATION.

VERIFICATION CONDITION GENERATOR (VCG)
A SOFTWARE PACKAGE USUALLY, BUT NOT NECESSARILY, AN INTERIM STAGE AS PART OF AN AUTOMATED PROOF OF CORRECTNESS PACKAGE THAT RECEIVES A PARSED PROGRAM/ ASSERTION AS INPUT, AND OUTPUTS A STRING OF THEOREMS THAT WILL BE INPUT TO THE THEOREM PROVER.

VERIFICATION TOOLS
VERIFICATION TOOLS ARE COMPUTERIZED AIDS THAT AUTOMATE PORTIONS OF THE ANALYSIS AND TESTING ACTIVITIES. (LISTING OF TYPES AND FUNCTIONS DAN306 P42)

VERIFICATION/VALIDATION/CERTIFICATION
VERIFICATION/VALIDATION/CERTIFICATION (OF COMPUTER PROGRAMS). THE PROCESS OF DETERMINING THAT THE COMPUTER PROGRAM WAS DEVELOPED IN ACCORDANCE WITH THE STATED SPECIFICATION AND SATISFACTORILY PERFORMS, IN THE MISSION ENVIRONMENT, THE FUNCTION(S) FOR WHICH IT WAS DESIGNED. SEE COMPUTER PROGRAM VERIFICATION, COMPUTER PROGRAM VALIDATION, COMPUTER PROGRAM CERTIFICATION. (DAN 134)

VERSION MODIFICATION LEVEL
AN INDICATION OF THE VERSION AND MODIFICATION LEVEL OF A UNIT OF SOURCE CODE. (DAN 137)

VIABILITY
VIABILITY IS DEFINED AS THE ADEQUACY WITH WHICH A GIVEN ALGORITHM MEETS TIMING CONSTRAINTS. (DAN 781)

VIRTUAL MACHINE MONITOR
THE PROGRAM WHICH MEDIATES BETWEEN THE VIRTUAL MACHINE AND THE ACTUAL RESOURCES OF THE SYSTEM. (HOST MACHINE)

VIRTUAL MACHINE(S)
A HARDWARE-SOFTWARE DUPLICATE OF A REAL EXISTING COMPUTER SYSTEM IN WHICH A STATISTICALLY DOMINANT SUBSET OF THE VIRTUAL PROCESSOR'S INSTRUCTIONS EXECUTE DIRECTLY ON THE HOST (OR REAL) PROCESSOR IN NATIVE MODE. (2) A TEST TOOL WHICH ALLOWS MULTIPLE SOFTWARE SYSTEMS TO EXECUTE ON ONE PHYSICAL MACHINE, BUT EACH SYSTEM THINKS THAT IT HAS SOLE ACCESS AND CONTROL OF A SINGLE DEDICATED PHYSICAL MACHINE. (DAN 286) (3) THE FUNCTIONAL EQUIVALENT OF A REAL MACHINE. (ANSI-X3H1) SEE ALSO: ABSTRACT MACHINE.

VIRTUAL MEMORY
COMPUTER STORAGE THAT APPEARS EXTERNALLY TO HAVE UNLIMITED CAPACITY.

WALK-THROUGH
FORMAL MEETING SESSIONS FOR THE REVIEW OF SOURCE CODE AND DESIGN BY THE VARIOUS MEMBERS OF THE PROJECT, FOR TECHNICAL RATHER THAN MANAGEMENT PURPOSES. THE PURPOSE IS FOR ERROR DETECTION AND NOT CORRECTION. (SEL) (2) A FORMAL, MULTIDISCIPLINARY PAPER DESIGN REVIEW OF A COMPUTER PROGRAM, SPECIFICATION, STRUCTURE, PROGRAM, LOGIC, MODULES, CODE,ETC., OFTEN USING HYPOTHETICAL INPUTS. (NASA)

WEIBULL DISTRIBUTION
INDEXING TERM. REFERS TO THE MATHEMATICAL METHODOLOGY WHICH IS USED TO CONSTRUCT, OR WHICH IS THE FORM ASSUMED BY, A PARTICULAR MODEL.

WELLMADE
A SOFTWARE DESIGN METHODOLOGY WHICH SEEKS TO DERIVE A PROVABLY CORRECT PROGRAM FROM THE FUNCTIONAL SPECIFICATIONS. (DAN 254)

WHILE
PROGRAM CONTROL CONSTRUCT WHERE CONTROL STAYS AS LONG AS THE CONTROL

VARIABLES SATISFY THE SPECIFIED CONDITION... THE WHILE CONSTRUCTION IS CONSIDERED AS A "GOTO" REPLACEMENT. PROGRAMMING LANGUAGE BLISS IS A "GOTO-LESS" LANGUAGE BUT CONTAINS A "WHILE-DO" CONSTRUCT. (SET)

WORD
A SEQUENCE OF A PARTICULAR LENGTH OF 0'S AND 1'S WHICH IS INTERPRETED BY A COMPUTER AS AN INSTRUCTION OR ELEMENT OF DATA. (NASA)

WORK BREAKDOWN STRUCTURE
A MANAGEMENT TOOL USED ON PROJECTS AND PROPOSALS TO IDENTIFY THE INDIVIDUAL COST CENTERS FOR ASSIGNING COSTS AND MAINTAINING CONTROL ON ALL INDIVIDUAL ITEMS OF WORK ON A PROJECT. (DAN 1201) (2) AN ENUMERATION OF ALL WORK ACTIVITIES IN HIERARCHIC REFINEMENTS OF DETAIL THAT DEFINES WORK TO BE DONE INTO SHORT, MANAGEABLE TASKS WITH QUANTIFIABLE INPUTS, OUTPUTS, SCHEDULES, AND ASSIGNED RESPONSIBILITIES. IT IS USED FOR PROJECT BUDGETING OF TIME AND RESOURCES DOWN TO THE INDIVIDUAL TASK LEVEL, AND AS A BASIS FOR PROGRESS REPORTING RELATIVE TO MEANINGFUL MANAGEMENT MILESTONES. (DAN 1153)

WORKAROUND
THE METHOD USED TO COUNTERACT THE EFFECTS OF AN ERROR IN A PROGRAM WHEN THE CAUSE OF THE ERROR, AND CONSEQUENTLY THE LOCATION OF THE STATEMENTS CONTAINING THE ERROR, IS NOT KNOWN. (SEL)

WORKING SETS
A PROGRAM'S WORKING SET IS A COLLECTION OF RECENTLY REFERENCED PAGES (OR SEGMENTS) OF A PROGRAM'S VIRTUAL ADDRESS SPACE. BECAUSE IT IS SPECIFIED IN THE PROGRAM'S VIRTUAL TIME, THE WORKING SET PROVIDES AN INTRINSIC MEASUREMENT OF THE PROGRAM'S MEMORY DEMAND-- I.E. A MEASUREMENT THAT IS UNPERTURBED BY ANY OTHER PROGRAM IN THE SYSTEM OR BY THE MEASUREMENT PROCEDURE ITESELF.

ZIPF'S LAWS
A LINGUISTIC THEORY WHICH DESCRIBES THE STRUCTURE OF A WRITTEN OR SPOKEN NATURAL LANGUAGE. IN PARTICULAR, THIS THEORY IS BEING EXTENDED TO PROGRAMMING LANGUAGES. (DAN 297).

130 blank

SECTION III

SOURCES

3.1 *Bibliography of Sources*

DAN 14 OGDIN, JERRY L., 'IMPROVING SOFTWARE RELIABILITY', DATAMATION, JANUARY 1973, PP. 49-52.

DAN 21 CRAIG, G.R., HETRICK, W.L., LIPOW, M., THAYER, T.A., ET AL, 'SOFTWARE RELIABILITY STUDY', RADC-TR-74-250, OCTOBER 1974, 110P. AVAIL:NTIS (ORDER NUMBER AD-787-784)

DAN 31 SHOOMAN, MARTIN L., 'SOFTWARE RELIABILITY: MEASUREMENT AND MODELS', PROCEEDINGS 1975 ANNUAL RELIABILITY AND MAINTAINABILITY SYMPOSIUM, JAN. 1975, PP. 485-591.

DAN 109 ROSS, DOUGLAS T., GOODENOUGH, JOHN B., IRVINE,C.A., 'SOFTWARE ENGINEERING: PROCESS, PRINCIPLES, AND GOALS', COMPUTER, MAY 1975, PP. 17-27.

DAN 132 BOEHM, B.W., BROWN, J.R., KASPER, H., LIPOW, M., MACLEOD, G.J. MERRITT, M.J. 'CHARACTERISTICS OF SOFTWARE QUALITY', TRW-SS-73-09, 28 DECEMBER 1973, 166P. AVAIL:TRWD

DAN 134 REIFER, D.J., 'INTERIM REPORT ON THE AIDS INVENTORY PROJECT', REPORT SAMSO-TR-75-184, 16 JULY 1975, 70P. AVAIL: NTIS

DAN 136 TRIMBLE, JOHN T. 'STRUCTURED PROGRAMMING SERIES VOLUME IV, DATA STRUCTURING STUDY', 21 APRIL 1975, RADC-TR-74-300, 46P. AVAIL:NTIS

DAN 137 LUPPINO, F.M. ET.AL., 'STRUCTURED PROGRAMMING SERIES, VOLUME V, PROGRAMMING SUPPORT LIBRARY (PSL) FUNCTIONAL REQUIREMENTS', 25 JULY 1974, RADC -TR-74-300, 55P. AVAIL:NTIS (ORDER NUMBER AD/A-003 339)

DAN 140 BARRY, BARBARA S., 'STRUCTURED PROGRAMMING SERIES, VOLUME X, CHIEF PROGRAMMER TEAM OPERATIONS DESCRIPTION', 22 JANUARY 1975, RADC-TR-74-300, 51P. AVAIL:NTIS (ORDER NUMBER AD-A008 861)

DAN 141 SMITH, RONALD L., 'STRUCTURED PROGRAMMING SERIES, VOLUME IX, MANAGEMENT DATA COLLECTION AND REPORTING' 24 OCTOBER 1974, RADC-TR-74-300, 128P. AVAIL:NTIS (ORDER NUMBER AD-A008-640)

DAN 142 KESSLER, MARVIN M., KISTER, WILLIAM E., 'STRUCTURED PROGRAMMING SERIES VOLUME XIV, SOFTWARE TOOL IMPACT', 22 MAY 1975, RADC-TR-74-300, 29P. AVAIL:NTIS

DAN 154 SMITH, RONALD L., 'STRUCTURED PROGRAMMING SERIES VOLUME XV, VALIDATION AND VERIFICATION STUDY, MAY 1975, RADC-TR-74-300, 82P. AVAIL:NTIS

DAN 158 MANLEY, JOHN JH., 'EMBEDDED COMPUTER SYSTEM SOFTWARE RELIABILITY', DEFENSE MANAGEMENT JOURNAL, VOL. 11, NO 4, OCTOBER 1975, PP. 13-17.

DAN 172 CLAPP, J.A., 'SOFTWARE ENGINEERING: PROBLEMS AND FUTURE DEVELOPMENTS', NOVEMBER 1974, MTR-2791, ESD-TR-74-195

DAN 209 MCCABE, THOMAS J., 'A COMPLEXITY MEASURE', IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-2, NO. 4, DEC. 1976, PP. 308-320.

DAN 210 MORANDA, PAUL B., 'QUANTITATIVE METHODS FOR SOFTWARE RELIABILITY MEASUREMENTS', TECNICAL REPORT AFOSR-TR-77-0046, DECEMBER 1976, 191P. AVAIL:NTIS (ORDER NUMBER ADA035585)

DAN 222 JOINT TECHNICAL COORDINATING GROUP ON ELECTRONICS EQUIPMENT RELIABILITY, 'SUMMARY REPORT OF THE JOINT LOGISTICS COMMANDERS ELECTRONICS SYSTEMS RELIABILITY WORKSHOP', AUGUST 1975, 37P. AVAIL:NTIS (ORDER NUMBER AD-A014 568)

DAN 223 FIFE, DENNIS W., 'COMPUTER SOFTWARE MANAGEMENT: A PRIMER FOR PROJECT MANAGEMENT AND QUALITY CONTROL', NBS SPECIAL PUBLICATIONS; 500-11,JULY 1977, 58P. AVAIL:GPO

DAN 225 AEROSPACE CORPORATION, 'FAULT-TOLERANT SOFTWARE FOR AIRCRAFT CONTROL SYSTEMS', TECHNICAL REPORT NO. ATR-78(7640)-1 FEB 1978, 74P. AVAIL:NTIS

DAN 226 HECHT, H., ET.AL, 'RELIABILITY MEASUREMENT DURING SOFTWARE DEVELOPMENT', NASA CR-145205, SEPTEMBER 1977, 96P. AVAIL:NTIS

DAN 227 MYERS, WARE, 'THE NEED FOR SOFTWARE ENGINEERING', COMPUTER, FEBRUARY 1978, PP. 12-26.

DAN 230 THAYER, R.H., AND LEHMAN, J.H., 'SOFTWARE ENGINEERING PROJECT MANAGEMENT: A SURVEY CONCERNING U.S. AEROSPACE INDUSTRY MANAGEMENT OF SOFTWARE DEVELOPMENT PROJECTS', REPORT NO. SM-ALC/ADC-TR-77-02, 1977, 19P. (SUMMARY APPEARS IN PROCEEDINGS OF AIAA CONFERENCE 'COMPUTERS IN AEROSPACE', NOVEMBER 1977)

DAN 231 FITZSIMMONS, A., AND LOVE, T., 'A REVIEW AND EVALUATION OF SOFTWARE SCIENCE', ACM COMPUTING SURVEYS, MARCH 1978,PP. 3-18.

DAN 232 SHOOMAN, M.L., AND RUSTON H., 'SOFTWARE MODELING STUDIES SUMMARY OF TECHNICAL PROGRESS', RADC-TR-78-4, VOLUME I, JANUARY 1978, 42P. AVAIL:NTIS

DAN 233 BROWN, J.R. AND NELSON, E.C., 'FUNCTIONAL PROGRAMMING', TECHNICAL REPORT, RADC-TR-78-24, FEB 1978, 101P. AVAIL:NTIS

DAN 234 BERNING, PAUL T., ANDERSON, ERIC R., AND BELZ, FRANK C., AUTOMATED COMPILER TEST CASE GENERATION, TECHNICAL REPORT, RADC-TR-78-30, FEB. 1978, 103P. AVAIL:NTIS

DAN 235 SUKERT, ALAN N., 'A FOUR-PROJECT EMPIRICAL STUDY OF SOFTWARE ERROR PREDICTION MODELS', DRAFT OF PAPER, 1978, 24P.

DAN 236 RANDALL, B., LEE, P.A., TRELEAVEN, P.C., 'RELIABILITY ISSUES IN COMPUTING SYSTEM DESIGN', ACM COMPUTING SURVEYS, VOL. 10, NO 2, JUNE 1978, PP. 123-165.

DAN 237 ZELKOWITZ, MARVIN G., 'PERSPECTIVES ON SOFTWARE ENGINEERING', ACM COMPUTING SURVEYS, VOL. 10, NO.2, JUNE 1978, PP. 197-216.

DAN 238 SCHICK, GEORGE J. AND WOLVERTON, RAY W., 'AN ANALYSIS OF COMPETING SOFTWARE RELIABILITY MODELS', IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-4, NO.2, MAR. 1978, PP. 104-120.

DAN 239 LLOYD, D.K., AND LIPOW, M., RELIABILITY MANAGEMENT, METHODS, AND MATHEMATICS, PUBLISHED BY THE AUTHORS, REDONDO BEACH, CA, 1977, 589P.

DAN 242 RIDDLE, WM. E., WILEDON, JACK C., SAYLOR, JOHN H., SEGAL, ALAN R., AND STAVELY, ALLAN M., 'BEHAVIOR MODELING DURING SOFTWARE DESIGN', PROCEEDINGS OF 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 13-22. AVAIL:IEES

DAN 243 KIEBURTZ, R.B., BARABASH, W., HILL, C.R., 'A TYPE-CHECKING PROGRAM LINKAGE SYSTEM FOR PASCAL', PROCEEDING OF 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 23-28. AVAIL:IEES

DAN 244 HAMILTON, PATRICIA A., MUSA, JOHN D., 'MEASURING RELIABILITY OF COMPUTATION CENTER SOFTWARE', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 29-36. AVAIL:IEES

DAN 245 LITTLEWOOD, B., 'HOW TO MEASURE SOFTWARE RELIABILITY AND HOW NOT TO...', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 37-45. AVAIL:IEES

DAN 246 MIYAMOTO, ISAO, 'TOWARD AN EFFECTIVE SOFTWARE RELIABILITY EVALUATION', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 46-55. AVAIL:IEES

DAN 250 JACKSON, M.A., 'INFORMATION SYSTEMS: MODELING,SEQUENCING AND TRANSFORMATIONS', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 72-81. AVAIL:IEES

DAN 251 FISHER, DAVID A., 'THE INTERACTION BETWEEN THE PRELIMINARY DESIGNS AND THE TECHNICAL REQUIREMENTS FOR THE DOD COMMON HIGH ORDER LANGUAGE, PROCEEDINGS, 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 82-83. AVAIL:IEES

DAN 253 PEDERSON, JAN T., BUCKLE, JOHN K., 'KONGSBERG'S ROAD TO AN INDUSTRIAL SOFTWARE METHODOLOGY', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 85-93. AVAIL:IEES

DAN 254 BOYD, DONALD L., PIZZARELLO, ANTONIO, 'INTRODUCTION TO THE WELLMADE DESIGN METHODOLOGY', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE

ENGINEERING, MAY 1978, PP. 94-100. AVAIL:IEES

DAN 255 STEPHENS, SHARON A., TRIPP, LEONARD L., 'REQUIREMENTS EXPRESSION AND VERIFICATION AID', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 101-108. AVAIL:IEES

DAN 256 WILLIS, R.R., 'DAS - AN AUTOMATED SYSTEM TO SUPPORT DESIGN ANALYSIS', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 109-115. AVAIL:IEES

DAN 258 DNIESTROWSKI, A., GUILLAUME, J.M., AND MORTIER, R., 'SOFTWARE ENGINEERING IN AVIONICS APPLICATIONS', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 124-131. AVAIL:IEES

DAN 260 BROWN, JOHN R. AND FISCHER, KURT, F., 'A GRAPH THEORETIC APPROACH TO THE VERIFICATION OF PROGRAM STRUCTURES', PROCEEDINGS, 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 136-141. AVAIL:IEES

DAN 261 BROWNE, J.C. AND JOHNSON, DAVID B., 'FAST: A SECOND GENERATION PROGRAM ANALYSIS SYSTEM', PROCEEDINGS OF THE 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 142-148. AVAIL:IEES

DAN 262 MCCLURE, CARMA L., ' MODEL FOR PROGRAM COMPLEXITY ANALYSIS PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 149-157. AVAIL:IEES

DAN 263 DERSHOWITZ, NACHUM AND MANNA, ZOHAR, 'INFERENCE RULES FOR PROGRAM ANNOTATION', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 158-167. AVAIL:IEES

DAN 264 AZEMA, P., AYACHE, J.M., BERTHOMIEU, B., 'DESIGN AND VERIFICATION OF COMMUNICATION PROCEDURES: A BOTTOM-UP APPROACH PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 168-174. AVAIL:IEES

DAN 265 MANNA, ZOHAR AND WALDINGER, RICHARD, 'THE SYNTHESIS OF STRUCTURE-CHANGING PROGRAMS', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 175-187. AVAIL:IEES

DAN 269 BOI, M.M.L. AND MICHEL, P., 'DESIGN AND PRINCIPLES OF A FAULT TOLERANT SYSTEM', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 207-214. AVAIL:IEES

DAN 270 CHUNG, PAUL AND GAIMAN, BARRY, 'USE OF STATE DIAGRAMS TO ENGINEER COMMUNICATIONS SOFTWARE', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 215-221. AVAIL:IEES

DAN 271 SCOTT, LEIGHTON R., 'AN ENGINEERING METHODOLOGY FOR PRESENTING SOFTWARE FUNCTIONAL ARCHITECTURE', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 222-229. AVAIL:IEES

DAN 272 CAMPOS, IVAN M., AND ESTRIN, GERALD, 'CONCURRENT SOFTWARE SYSTEM DESIGN SUPPORTED BY SARA AT THE AGE OF ONE', PROCEEDINGS, 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 230-242. AVAIL:IEES

DAN 273 WEGNER, PETER, 'RESEARCH DIRECTIONS IN SOFTWARE TECHNOLOGY', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING', MAY 1978, PP. 243-259. AVAIL:IEES

DAN 275 PARNAS, DAVID L., 'DESIGNING SOFTWARE FOR EASE OF EXTENSION AND CONTRACTION', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 264-277. AVAIL:IEES

DAN 277 COOK, DOUGLAS, 'MEASURING MEMORY PROTECTION', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 281-287. AVAIL:IEES

DAN 278 ALMES, GUY, AND ROBERTSON, GEORGE, 'AN EXTENSIBLE FILE SYSTEM FOR HYDRA', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 288-294. AVAIL:IEES

DAN 279 GOULLON, HANNES; ISLE, RAINER; AND LOHR, KLAUS-PETER,'DYNAMIC RESTRUCTURING IN AN EXPERIMENTAL OPERATING SYSTEM', PROCEEDINGS, 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 230-242. AVAIL:IEES

DAN 281 PERSCH, GUIDO AND WINTERSTEIN, GEORGE, 'SYMBOLIC INTERPRETATION AND TRACING OF PASCAL-PROGRAMS', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 312-319. AVAIL:IEES

DAN 282 PANZL, DAVID J., 'AUTOMATIC REVISION OF FORMAL TEST PROCEDURES', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 320-326. AVAIL:IEES

DAN 283 STERN, MAX, 'SOME EXPERIENCE IN BUILDING PORTABLE SOFTWARE', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 327-332. AVAIL:IEES

DAN 284 THALMANN, DANIEL, EVOLUTION IN THE DESIGN OF ABSTRACT MACHINES FOR SOFTWARE PORTABILITY', PROCEEDINGS 3RD INT'L CONFERENCE ON SOFTWARE ENGINEERING, MAY 1978, PP. 333-340. AVAIL:IEES

DAN 286 MYERS, GLENFORD J., 'SOFTWARE RELIABILITY PRINCIPLES AND PRACTICES', WILEY INTERSCIENCE PUBLICATIONS, 1976, 360P.

DAN 288 SCHUSTER, S.A. AND REGHBATI, E., 'AN APPROACH TO THE ENGINEERING OF DATA PROCESSING PROGRAMS',PROCEEDINGS OF COMPSAC 77, NOV. 1977, PP. 540-546. AVAIL:IEES

DAN 289 LOCKS, MITCHELL O., 'MONTE CARLO BAYESIAN SYSTEM RELIABILITY AND MTBF - CONFIDENCE ASSESSMENT, 11', VOL. 1, 'THEORY', VOL. 2, 'SPARCS-2 USER'S MANUAL', TECHNICAL REPORT, MAR. 1978, 103P.

DAN 290 ISRAD,'U.S. ARMY INTEGRATED SOFTWARE RESEARCH AND DEVELOPMENT PROGRAM' VOL. 1 - EXECUTIVE SUMMARY, APRIL 1977, 20 P.

DAN 292 RYE, P., BAMBERGER, F., OSTANEK, W., BRODEUR, N., GOODE,J. 'SOFTWARE SYSTEMS DEVELOPMENT: A CSDL PROJECT HISTORY', JUNE 1977, 91P. FINAL TECHNICAL REPORT, RADC-TR-77-213. AVAIL:IEES

DAN 295 FIRES, M.J., 'SOFTWARE ERROR DATA ACQUISITION', FINAL TECHNICAL REPORT, RADC-TR-77-130, APRIL 1977. AVAIL:NTIS

DAN 296 GOEL, AMRIT L., AND OKUMOTO, K., 'BAYESIAN SOFTWARE PREDICTION MODELS, 4 VOLS., FINAL TECHNICAL REPORT RADC-TR-78-155, VOLS 1-4, JULY 1978. AVAIL:NTIS

DAN 297 LAEMMEL, A.L., AND SHOOMAN, M., 'SOFTWARE MODELING STUDIES STATISTICAL (NATURAL) LANGUAGE THEORY AND COMPUTER PROGRAM COMPLEXITY', FINAL TECHNICAL REPORT, RADC-TR-78-4, VOL. II, APR. 1977, 48P. AVAIL:NTIS

DAN 298 GOEL, AMRIT L., 'SUMMARY OF TECHNICAL PROGRESS ON BAYESIAN SOFTWARE PREDICTION MODELS', INTERIM REPORT, RADC-TR-77-112, MARCH 1977, 42P. AVAIL:NTIS

DAN 299 SHOOMAN, M.L. AND RUSTON, H., SUMMARY OF TECHNICAL PROGRESS, SOFTWARE MODELING STUDIES', INTERIM REPORT, RADC-TR- 77-88, MARCH 1977, 42P. AVAIL:NTIS

DAN 300 PRENTISS, NELSON H., JR., 'VIKING SOFTWARE DATA', TECHNICAL REPORT, RADC-TR-77-168, 1977, 293P. AVAIL:NTIS

DAN 301 UHRIG, J.L., 'LIFE-CYCLE', CYCLE EVALUATION OF SYSTEM PARTITIONING', PROCEEDINGS, COMPSAC 77, 1977, PP. 2-8. AVAIL:IEES

DAN 303 DEWOLF, BARTON J., 'REQUIREMENTS SPECIFICATION AND PRELIMINARY DESIGN FOR REAL-TIME SYSTEMS', PROCEEDINGS, COMPSAC 77, 1977, PP. 17-23. AVAIL:IEES

DAN 306 FUJII, MARILYN S., 'INDEPENDENT VERIFICATION OF HIGHLY RELIABLE PROGRAMS', PROCEEDINGS, COMPSAC 77, 1977, PP. 38-44. AVAIL:IEES

DAN 308 CHOW, TSUN S., 'TESTING SOFTWARE DESIGN MODELED BY FINITE STATE MACHINES', PROCEEDINGS, COMPSAC 77, 1977, PP. 58-64. AVAIL:IEES

DAN 311 CHAN, ALEX M., LUI, JAMES C., RODE, ELVA E., SLEKYS, ARUNAS G., 'TASC: A FAULT-TOLERANT MINICOMPUTER-BASED SYSTEM FOR CENTRALIZED MAINTENANCE OF ELECTRONIC SWITCHERS', PROCEEDINGS, COMPSAC 77, 1977, PP. 120-126. AVAIL:IEES

DAN 313 ZOLNOWSKI, JEAN M. AND SIMMONS, DICK B., 'A COMPLEXITY MEASURE APPLIED TO FORTRAN', PROCEEDINGS, COMPSAC 77, 1977, PP. 133-141. AVAIL:IEES

DAN 314 CHEN, EDWARD T., 'PROGRAM COMPLEXITY AND PROGRAMMER PRODUCTIVITY', PROCEEDINGS, COMPSAC 77, 1977, PP. 142-148. AVAIL:IEES

DAN 315 AVIZIENIS, ALGIRDAS AND CHEN, LIMING, '(N THE IMPLEMENTATION OF N-VERSION PROGRAMMING FOR SOFTWARE FAULT-TOLERANCE DURING PROGRAM EXECUTION', PROCEEDINGS, COMPSAC 77, 1977, PP. 149-155. AVAIL:IEES

DAN 318 IRVABUCHI, EISUKE; SOGA, KENTARO; AND KAWAI, YOICHI; 'STRUCTURED DESIGN OF ELECTRONIC SWITCHING SYSTEMS, PROCEEDINGS, COMPSAC 77,

1977, PP. 179-185. AVAIL:IEES

DAN 322 FERRENTINO, A.B., AND MILLS, H.D., 'STATE MACHINES AND THEIR SEMANTICS IN SOFTWARE ENGINEERING', PROCEEDINGS, COMPSAC 77, 1977, PP. 242-251. AVAIL:IEES

DAN 323 KODRES, UNO R., 'ANALYSIS OF REAL-TIME SYSTEMS BY DATA FLOWGRAPHS', PROCEEDINGS, COMPSAC 77, 1977, PP. 300-305. AVAIL:IEES

DAN 326 CAREY, ROBERT AND BENDICK, MARC, 'THE CONTROL OF A SOFTWARE TEST PROCESS', PROCEEDINGS, COMPSAC 77, 1977, PP. 327-333. AVAIL:IEES

DAN 327 STRAETER, TERRY A., FOUDRIAT, EDWIN C., AND WILL, RALPH W 'RESEARCH FLIGHT SOFTWARE ENGINEERING AND MUST, AN INTEGRATED SYSTEM OF SUPPORT TOOLS', PROCEEDINGS, COMPSAC 77, 1977, PP.392-396. AVAIL:IEES

DAN 333 CHESTER, DANIEL L., AND YEH, RAYMOND T., 'SOFTWARE DEVELOPMENT BY EVALUATION OF SYSTEM DESIGNS', PROCEEDINGS, COMPSAC 77, 1977, PP. 435-441. AVAIL:IEES

DAN 335 SHARPLEY, W.K., JR., 'SOFTWARE MAINTENANCE PLANNING FOR EMBEDDED COMPUTER SYSTEMS', PROCEEDINGS, COMPSAC 77, 1977, PP. 520-526. AVAIL:IEES

DAN 338 ELSHOFF, JAMES L., 'ON OPTIMAL MODULE SIZE WITH RESPECT TO COMPILATION COST', PROCEEDINGS, COMPSAC 77, 1977, PP. 547-553. AVAIL:IEES

DAN 346 DES JARDINS, RICHARD, 'EVOLUTIONARY DISTRIBUTED SYSTEMS DESIGN', PROCEEDINGS, COMPSAC 77, 1977, PP. 765-771. AVAIL:IEES

DAN 347 BEDARD, C.J., MELLOR, F., AND OLDER, W.J., 'A MESSAGE-SWITCHED OPERATING SYSTEM FOR A MULTIPROCESSOR', PROCEEDINGS, COMPSAC 77, 1977, PP. 772-777. AVAIL:IEES

DAN 355 TURN R., DAVIS, M.R., REINSTEDT, R.N., 'A MANAGEMENT APPROACH TO THE DEVELOPMENT OF COMPUTER-BASED SYSTEMS', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 11-17. AVAIL:AIAA

DAN 356 SYLVESTER, DR. RICHARD J., 'ELEMENTS OF THE COMPUTER PROGRAM DEVELOPMENT PLAN'. A COLLECTION OF TECHNICAL PAPERS COMPUTERS IN AEROSPACE CONFERENCE, NOV.1977, PP. 18-22. AVAIL:AIAA

DAN 360 OSTERWEIL, LEON J., 'A METHODOLOGY FOR TESTING COMPUTER PROGRAMS', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 52-62. AVAIL:AIAA

DAN 361 GODOY, S.G., AND ENGELS, G.J.,'SOFTWARE SNEAK ANALYSIS', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 63-67. AVAIL:AIAA

DAN 366 BATE, ROGER R., 'SOFTWARE DESIGN PROCEDURES', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 102-107. AVAIL:AIAA

DAN 370 WHEATLEY, RICHARD, 'A SIMULATION TECHNIQUE IN MICROPROGRAM VALIDATION', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 125-129. AVAIL:AIAA

DAN 381 TERNANDEZ, MANUEL, 'A REVIEW OF DOD AND NASA COMPUTER STANDARDIZATION', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 232-239. AVAIL:IEES

DAN 382 THOMPSON, C.H., 'OVERVIEW OF AIR FORCE LOGISTICS SOFTWARE MANAGEMENT', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 257-259. AVAIL:AIAA

DAN 384 JELINSKI, Z., 'DECREASING DESIGN ERRORS AND PROBLEMS WITH SUPPORT SOFTWARE AS DELIVERABLES', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 268-275. AVAIL:AIAA

DAN 385 FOX, COL. LOREN J., 'E-3A SOFTWARE MAINTENANCE',A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 276-285. AVAIL:AIAA

DAN 388 MARTIN, DR. FRED H., 'HAL/S - THE AVIONICS PROGRAMMING SYSTEM FOR SHUTTLE', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 308-318. AVAIL:AIAA

DAN 389 BONNEAU, R.J. 'IMPROVED OPERATING SYSTEMS RELIABILITY THROUGH LANGUAGE FEATURES', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 319-324. AVAIL:AIAA

DAN 390 SMITH, K.V. AND JELINSKI, Z., 'HOLDET HIGHER LANGUAGE EVALUATION TOOL', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 325-328. AVAIL:AIAA

DAN 393 C. CANNON, 'A VERIFICATION CASE STUDY', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 349-353. AVAIL:AIAA

DAN 402 SUKERT, ALAN N., 'A MULTI-PROJECT COMPARISON OF SOFTWARE RELIABILITY MODELS', A COLLECTION OF TECHNICAL PAPERS, COMPUTERS IN AEROSPACE CONFERENCE, NOV. 1977, PP. 413-421. AVAIL:AIAA

DAN 412 YAO, S.B. BERNSTEIN, PHILIP A., GOODMAN, NATHAN, SCHUSTER, STEWART A., SHIPMAN, DAVID, AND SMITH, DIANE C.P.,'DATA-BASE SYSTEMS', COMPUTER, SEPT. 1978, VOL. 11, NO. 9, PP. 46-60.

DAN 415 KING, JOHN LESLIE AND SCHREMS, EDWARD L. 'COST-BENEFIT ANALYSIS IN INFORMATION SYSTEMS DEVELOPMENT AND OPERATION', ACM COMPUTING SURVEYS, MAR. 1978, VOL. 10, NO. 1, PP. 19-34.

DAN 419 MANNA, ZOHAR AND WALDINGER, RICHARD, 'IS SOMETIME SOMETIMES BETTER THAN ALWAYS?', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 32-39. AVAIL:IEES

DAN 420 KARP, RICHARD ALAN AND LUCKHAM, DAVID C., 'VERIFICATION OF

FAIRNESS IN AN IMPLEMENTATION OF MONITORS', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 40-46. AVAIL:IEES

DAN 421 HOWARD, JOHN H., 'SIGNALING IN MONITORS', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 47-52. AVAIL:IEES

DAN 422 SAXENA, ASHOK R., AND BREDT, THOMAS H., 'VERIFICATION OF A MONITOR SPECIFICATION', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 53-60. AVAIL:IEES

DAN 423 BELL, T.E., AND THAYER, T.A., 'SOFTWARE REQUIREMENTS: ARE THEY REALLY A PROBLEM?', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 61-68. AVAIL:IEES

DAN 428 COOPER, DENNIS W., 'ADAPTIVE TESTING', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 102-105. AVAIL:IEES

DAN 434 BROWNE, J.C., 'A CRITICAL OVERVIEW OF COMPUTER PERFORMANCE EVALUATION', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 138-145. AVAIL:IEES

DAN 435 FERRARI, DOMENICO AND LAU, EDWIN, 'AN EXPERIMENT IN PROGRAM RESTRUCTURING FOR PERFORMANCE ENHANCEMENT', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 146-150. AVAIL:IEES

DAN 437 ZELKOWITZ, MARVIN V., 'AUTOMATIC PROGRAM ANALYSIS AND EVALUATION', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT 1976, PP. 146-150. AVAIL:IEES

DAN 448 FELDMAN, MICHAEL B., 'NEW LANGUAGES FROM OLD: THE EXTENSION OF PROGRAMMING LANGUAGES BY EMBEDDING, WITH A CASE STUDY', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 237-242. AVAIL:IEES

DAN 451 GOUDA, MOHAMED G., AND MANNING, ERIC G., 'ON THE MODELING, ANALYSIS AND DESIGN OF PROTOCOLS - A SPECIAL CLASS OF SOFTWARE STRUCTURES', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 256-262. AVAIL:IEES

DAN 458 TURN, R., DAVIS, M.R., AND REINSTEDT, R.N., 'A MANAGEMENT APPROACH TO THE DEVELOPMENT OF COMPUTER-BASED SYSTEMS', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 305-311. AVAIL:IEES

DAN 459 STEPHENSON, W.E., 'AN ANALYSIS OF THE RESOURCES USED IN THE SAFEGUARD SYSTEM SOFTWARE DEVELOPMENT', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 312-321. AVAIL:IEES

DAN 470 DENNING, PETER J., 'SACRIFICING THE CALF OF FLEXIBILITY ON THE ALTAR OF RELIABILITY', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976. PP. 384-386. AVAIL:IEES

DAN 476 YAU, S.S., CHEUNG, R.C., COCHRANE, D.C., 'AN APPROACH TO ERROR-RESISTANT SOFTWARE DESIGN', PROCEEDINGS, 2ND INT'L CONFERENCE ON SOFTWARE ENGINEERING, OCT. 1976, PP. 429-436. AVAIL:IEES

DAN 503 DUVALL, LORRAINE M., 'SOFTWARE DATA REPOSITORY STUDY TECHNICAL REPORT, RADC-TR-76-387, DEC. 1976, 153P. AVAIL:NTIS

DAN 509 THIBODEAU, ROBERT, 'THE STATE OF THE ART IN SOFTWARE ERROR DATA COLLECTION AND ANALYSIS, TECHNICAL REPORT', 1978, 115P. AVAIL:DACS

DAN 529 CICU, A., MAIOCCHI, M., POLILLO, R., SARDONI, A., 'ORGANIZING TESTS DURING SOFTWARE EVOLUTION', PROCEEDINGS OF 1975 CONFERENCE ON RELIABLE SOFTWARE, PP. 43-50. AVAIL:IEES

DAN 532 LISKOV, B.H., AND ZILLES, S., 'SPECIFICATION TECHNIQUES FOR DATA ABSTRACTIONS', PROCEEDINGS OF 1975 CONFERENCE ON RELIABLE SOFTWARE, PP. 72-87. AVAIL:IEES

DAN 535 BOEHM, B.W., MCCLEAN, R.K., AND URFRIG, D.B., 'SOME EXPERIENCES WITH AUTOMATED AIDS TO THE DESIGN OF LARGE-SCALE RELIABLE SOFTWARE', PROCEEDINGS OF 1975 CONFERENCE ON RELIABLE SOFTWARE, PP. 105-113. AVAIL:IEES

DAN 559 ENDRES, A., 'AN ANALYSIS OF ERRORS AND THEIR CAUSES IN SYSTEM PROGRAMS' PROCEEDINGS OF 1975 CONFERENCE ON RELIABLE SOFTWARE, PP. 327-336. AVAIL:IEES (ALSO IN IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, JUNE 1975, VOL. SE-1, NO. 2)

DAN 595 DENNING, PETER J., 'WORKING SETS TODAY', PROCEEDINGS, COMPSAC 78, 1978, PP. 71-77. AVAIL:IEES

DAN 609 HALLIN, T.C., AND HANSEN, R.C., 'TOWARD A BETTER METHOD OF SOFTWARE TESTING', PROCEEDINGS, COMPSAC 78, 1978, PP. 153-157. AVAIL:IEES

DAN 612 CHOW, T.S., 'ANALYSIS OF SOFTWARE DESIGN MODELED BY MULTIPLE FINITE STATE MACHINES', PROCEEDINGS OF COMPSAC 1978, PP. 169-174. AVAIL:IEES

DAN 616 DEMILLO, RICHARD A., AND DOBKIN, DAVID, 'RECENT PROGRESS IN SECURE COMPUTATION', PROCEEDINGS OF COMPSAC 1978, PP. 209-214. AVAIL:IEES

DAN 617 DENNING, DOROTHY E., 'A METHOD FOR MAINTAINING ROUTING DATA IN AUTOMATED RECORD KEEPING SYSTEMS', PROCEEDINGS OF COMPSAC 1978, PP. 215-219. AVAIL:IEES

DAN 668 YEE, JOHN G., AND SU, STEPHEN Y.H., 'A SCHEME FOR TOLERATING FAULTY DATA IN REAL-TIME SYSTEMS', PROCEEDINGS OF COMPSAC 1978, PP. 663-667. AVAIL:IEES

DAN 719 KODRES, UNO R., 'DISCRETE SYSTEMS AND FLOWCHARTS' IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-4, NO. 6, NOV. 1978, PP. 521-525.

DAN 720 RICH, CHARLES AND SHROBE, HOWARD, 'INITIAL REPORT ON A LISP PROGRAMMER'S APPRENTICE', IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-4, NO. 6, NOV. 1978, PP. 456-467.

DAN 724 MYERS, WARE, 'A STATISTICAL APPROACH TO SCHEDULING SOFTWARE

DEVELOPMENT', COMPUTER, VOL. 11, NO. 12, DEC. 1978, PP. 23-35.

DAN 737 FAIRLEY, RICHARD E., 'MODERN SOFTWARE DESIGN TECHNIQUES', PROCEEDINGS OF THE SYMPOSIUM OF COMPUTER SOFTWARE ENGINEERING, MICROWAVE RESEARCH INSTITUTE, SYMPOSIA SERIES VOL. XXIV, 1976, PP. 11-30. AVAIL:NYPP

DAN 748 AMSTER, S.J., DAVIS, E.J. DICKMAN, B.N., AND KUONI, J.P., 'AN EXPERIMENT IN AUTOMATIC QUALITY EVALUATION OF SOFTWARE' PROCEEDINGS OF THE SYMPOSIUM ON COMPUTER SOFTWARE ENGINEERING, MICROWAVE RESEARCH INSTITUTE, SYMPOSIA SERIES VOL. XXIV, 1976, PP. 171-197. AVAIL:NYPP

DAN 749 CURRY, R.W., ' MEASURE TO SUPPORT CALIBRATION AND BALANCING OF THE EFFECTIVENESS OF SOFTWARE ENGINEERING TOOLS AND TECHNIQUES', PROCEEDINGS OF THE SYMPOSIUM ON COMPUTER SOFTWARE ENGINEERING, MICROWAVE RESEARCH INSTITUTE, SYMPOSIA SERIES, VOL. XXIV, 1976, PP. 199-214. AVAIL:NYPP

DAN 753 YELOWITZ, L., 'SPECIFICATIONS, REFINEMENT, AND PROOF OF A MICROPROCESSOR', PROCEEDINGS OF THE SYMPOSIUM ON COMPUTER SOFTWARE EINGINEERING, MICROWAVE RESEARCH INSTITUTE, SYMPOSIA SERIES VOL. XXIV, 1976, PP. 251-266. AVAIL:NYPP

DAN 758 RAMAMOORTHY, C.V., AND JAHANIAN, P., 'FORMALIZING THE SPECIFICATION OF TARGET MACHINES FOR COMPILER ADAPTABILITY ENHANCEMENT', PROCEEDINGS OF THE SYPOSIUM ON COMPUTER SOFTWARE ENGINEERING, MICROWAVE RESEARCH INSTITUTE, SYMPOSIA SERIES, VOL. XXIV, 1976, PP. 353-366. AVAIL:NYPP

DAN 761 JOHNSON, J.N., AND SHAW, J.L., 'FAULT-TOLERANT SOFTWARE FOR A DUAL PROCESSOR WITH MONITOR', PROCEEDINGS OF THE SYMPOSIUM ON COMPUTER SOFTWARE ENGINEERING, MICROWAVE RESEARCH INSTITUTE, SYMPOSIA SERIES, VOL. XXIV, 1976, 395-407. AVAIL:NYPP

DAN 766 MOHANTY, S.N., AND ADAMOWICZ, M. 'PROPOSED MEASURES FOR THE EVALUATION OF SOFTWARE', PROCEEDINGS OF THE SYMPOSIUM ON COMPUTER SOFTWARE ENGINEERING, MICROWAVE RESEARCH INSTITUTE, SYMPOSIA SERIES, VOL. XXIV, 1976, PP. 485-497. AVAIL:NYPP

DAN 772 BOEHM, BARRY W., 'THE HIGH COST OF SOFTWARE', PAPER PRESENTED AT USC SEMINAR: PRACTICAL STRATEGIES FOR DEVELOPING LARGE SOFTWARE SYSTEMS; ADDISON WESLEY PUBL. CO., 1975, PP. 3-14.

DAN 773 SCHWARTZ, JULES I., 'CONSTRUCTION OF SOFTWARE: PROBLEMS AND PRACTICALITIES', PAPER PRESENTED AT USC SEMINAR: PRACTICAL STRATEGIES FOR DEVELOPING LARGE SOFTWARE SYSTEMS; ADDISON WESLEY PUBL. CO., 1975, PP. 15-53.

DAN 781 GILB, TOM, 'SOFTWARE METRICS', WINTHROP PUBLISHERS INC., 1977, 282P.

DAN 786 CARTER, LT. EDWARD M., 'DATA ELEMENT DICTIONARY/DIRECTORY SYSTEMS AND THE CONVERSION PROCESS', PROCEEDINGS OF THE ANNUAL COMPUTER RELATED INFORMATION SYSTEMS SYMPOSIUM, 1978, 20P. AVAIL:AFAA

DAN 813 KRALY, T.M., NAUGHTON J.J., SMITH, R.L. AND TINANOFF, N., 'PROGRAM DESIGN STUDY', STRUCTURED PROGRAMMING SERIES, VOL. VIII, TECHNICAL

REPORT RADC-TR-74-300, 1975, 66P. AVAIL:NTIS

DAN 837 DUNCAN, ARTHUR G., 'TEST GRAMMARS: A METHOD FOR GENERATING PROGRAM TEST DATA', DIGEST FOR THE WORKSHOP ON SOFTWARE TESTING AND TEST DOCUMENTATION, DEC. 1978, PP. 270-283.

DAN 841 BURNS, JAMES E., 'STABILITY OF TEST DATA FROM PROGRAM MUTATION', DIGEST FOR THE WORKSHOP ON SOFTWARE TESTING AND TEST DOCUMENTATION, DEC. 1978, PP. 324-334.

DAN 842 WHITE, LEE J. AND COHEN, EDWARD I., 'A DOMAIN STRATEGY FOR COMPUTER PROGRAM TESTING', DIGEST FOR THE WORKSHOP ON SOFTWARE TESTING AND TEST DOCUMENTATION, DEC. 1978, PP. 335-354.

DAN 843 LIPTON, RICHARD, AND SAYWARD, FREDERICK G., 'THE STATUS OF RESEARCH ON PROGRAM MUTATION', DIGEST FOR THE WORKSHOP ON SOFTWARE TESTING AND TEST DOCUMENTATION, DEC. 1978, PP. 355-373.

DAN 871 WOODWARD, M.R., HENNELL, M.A., AND HEDLEY, D., 'A MEASURE OF CONTROL FLOW COMPLEXITY IN PROGRAM TEST', IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-5, NO. 1, JAN. 1979, PP. 45-50.

DAN 872 CLARK, DOUGLAS W., 'MEASUREMENTS OF DYNAMIC LIST STRUCTURE USE IN LISP', IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-5, NO. 1, JAN. 1979, PP. 51-59.

DAN 874 BURNS, G., AND COHEN, B., 'TOWARDS A CONTRACTUAL METHODOLOGY', PAPER PRESENTED AT A WORKSHOP ON SOFTWARE TESTING AND TEST DOCUMENTATION, 8P. (COPY OF PAPER RECEIVED FROM AUTHOR - AWAITING REST OF CITATION)

DAN 1127 SHOLL, HOWARD A., AND BOOTH, TAYLOR A., 'SOFTWARE PERFORMANCE MODELLING USING COMPUTATION STRUCTURES', PROCEEDINGS OF THE 1ST NAT'L CONFERENCE ON SOFTWARE ENGINEERING, SEP. 1975, PP. 82-88. AVAIL:IEES

DAN 1153 TAUSWORTHE, ROBERT C., 'STANDARDIZED DEVELOPMENT OF COMPUTER SOFTWARE, PART II STANDARDS', TECHNICAL REPORT, AUG. 1977, 547P. AVAIL:NTIS

DAN 1201 BRANNING, W.E., SCHAENZER, J.P., WILLSON, D.M., ERICKSON, W.A.; 'MODERN PROGRAMMING PRACTICES STUDY REPORT', TECHNICAL REPORT, RADC-TR-77-106, APR. 1977, 419P. AVAIL:NTIS

DAN 1237 FIFE, DENNIS W., 'SOFTWARE MANAGEMENT STANDARDS', SOFTWARE PHENOMENOLOGY, WORKING PAPERS OF THE SOFTWARE LIFE CYCLE MANAGEMENT WORKSHOP, AUG. 1977, PP. 63-80. AVAIL:DDC

(ABBOTT) DEFINITION TAKEN FROM LIST COMPILED BY RUSSELL J. ABBOTT, DEPT. OF COMPUTER SCIENCE, CALIFORNIA STATE UNIVERSITY AT NORTHRIDGE, 18111 NORDHOFF STREET, NORTHRIDGE, CA 91330

(ANSI-X3) THIS MATERIAL IS REPRODUCED WITH PERMISSION FROM AMERICAN NATIONAL STANDARDS COMMITTEE X3 TECHNICAL REPORT AMERICAN NATIONAL DICTIONARY FOR INFORMATION PROCESSING, X3/TR-1-77, COPYRIGHT 1977 BY THE COMPUTER AND

BUSINESS EQUIPMENT MANUFACTURERS ASSOCIATION (CBEMA), COPIES OF WHICH MAY BE PURCHASED FROM THE AMERICAN NATIONAL STANDARDS INSTITUTE, 1430 BROADWAY, NEW YORK, NY 10018

(ANSI-X3H1) AMERICAN NATIONAL STANDARDS INSTITUTE, STANDING COMMITTEE ON OPERATING SYSTEM COMMAND LANGUAGES. DRAFT OF 21 MAY 1979. COMMITTEE CHAIRED BY LOIS C. FRAMPTON, DIGITAL EQUIPMENT CORP., 146 MAIN ST., MAYNARD, MA 01754. CURRENT VERSION MAINTAINED BY STEVEN MELLOR, U. OF CALIFORNIA, LAWRENCE BERKELEY LABORATORY, BLDG 46A, BERKELEY, CA 94720

(NASA) LOCKHEED, GEORGIA, 'INDUSTRY PERSPECTIVE ON SIMULATION METHODS AND RESEARCH FOR VALIDATION AND FAILURE EFFECTS. ANALYSIS OF ADVANCED DIGITAL FLIGHT CONTROL/AVIONICS'. TECHNICAL REPORT, NASA CR-152234, FEB. 1979, 300+P. LIMITED CIRCULATION - CONTROLLED DISTRIBUTION, NASA AMES RESEARCH CENTER, MOFFITT FIELD, CA 94035

(AFR-800-14) DEPARTMENT OF THE AIR FORCE, WASHINGTON, DC ACQUISITION AND SUPPORT PROCEDURES FOR COMPUTER RESOURCES IN SYSTEMS. AFR-800-14, VOL. 11, 26 SEPTEMBER 1975, 44P.

P730/D5 IEEE COMPUTER SOCIETY, TECHNICAL COMMITTEE, SUBCOMMITTEE ON SOFTWARE ENGINEERING STANDARDS, 'TRIAL-USE STANDARD FOR SOFTWARE QUALITY ASSURANCE PLANS', DEC. 1978, 13P.

(SET) SUBCOMMITTEE ON SOFTWARE ENGINEERING STANDARDS, TECHNICAL COMMITTEE ON SOFTWARE ENGINEERING, IEEE COMPUTER SOCIETY. 'SOFTWARE ENGINEERING TERMINOLOGY - DRAFT OF 23 MARCH 1978', ROBERT POSTON, H. HECHT, EDITORS, 63P.

(SEL) SOFTWARE ENGINEERING LABORATORY, SOFTWARE ENGINEERING LABORATORY GLOSSARY', NASA GODDARD SPACE FLIGHT CENTER, GREENBELT, MD 20770, 1978, 7P.

(LD4) WAGONER, W.L., 'THE FINAL REPORT ON A SOFTWARE RELIABILITY-MEASUREMENT STUDY', REPORT NO. TOP-0074(4112)-1, 15 AUGUST 1973, 54P. LIMITED DISTRIBUTION BY SAMSO/DYT, LOS ANGELES AIR FORCE STATION, LOS ANGELES, CA

(LD7) BRATMAN, HARVEY, CUDNEY, PAUL, AND JOHNSON, BRUCE; 'PROGRAM PRODUCTION LIBRARY PROGRAMMER'S GUIDE', SYSTEM DEVELOPMENT CORP. TM-5175-600-00, 10 AUGUST 1973, 56P. LIMITED DISTRIBUTION.

146 Blank

3.2 Ordering Information for Sources

CODE	NAME AND ADDRESS
ACM	ASSOCIATION FOR COMPUTING MACHINERY, INC., P.O. BOX 12105 CHURCH ST. STATION, NEW YORK, NY 10249
AFAA	U.S. AIR FORCE ACADEMY, DEPT. OF ASTRONAUTICS AND COMPUTER SCIENCE, DENVER, CO 80840
AIAA	AMERICAN INSTITUTE OF AERONAUTICS AND ASTRONAUTICS, 1290 AVENUE OF THE AMERICAS, NEW YORK, NY 10010
BELW	BELL LABORATORIES, WHIPPANY, NJ 07981
DACS	DATA AND ANALYSIS CENTER FOR SOFTWARE, RADC/ISISI, GRIFFISS AFB, ROME, NY 13441
DDC	DEFENSE DOCUMENTATION CENTER, CAMERON STATION, ALEXANDRIA, VA 22314
GPO	U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402
IEES	IEEE SERVICE CENTER, 445 HOES LANE, PISCATAWAY, NJ 08854
NTIS	NATIONAL TECHNICAL INFORMATION SERVICE, 5285 PORT ROYAL RD., SPRINGFIELD, VA 22161
NYPP	POLYTECHNIC PRESS OF THE POLYTECHNIC INSTITUTE OF NY, BROOKLYN, NY 11201
TRWD	TRW DEFENSE AND SPACE SYSTEMS GROUP, REDONDO BEACH, CA 90278


END
DATE
FILMED
B-84
DTIC